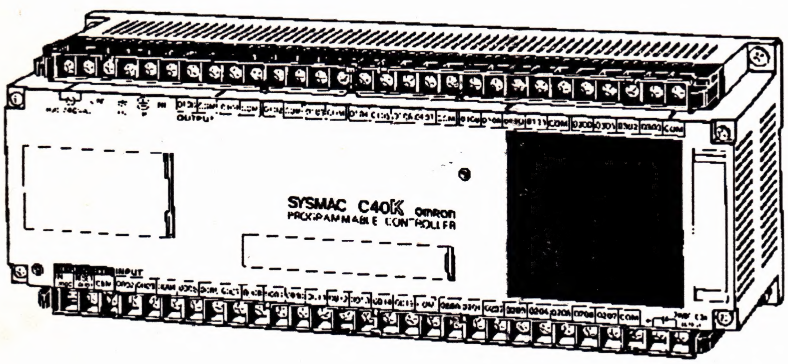
OMRON®

Basic Applications Manual

SYSMAC C20K/C28K/C40K Programmable Controllers



C20K BASIC APPLICATIONS
TABLE OF CONTENTS

SECTION		PAGE
1.0	INTRODUCTION	3
	1.1 COURSE DESCRIPTION	
	1.2 INTRODUCTION TO PROGRAMMABLE CONTROLLERS	
	1.3 C20K FAMILY SYSTEM COMPONENTS	
	1.4 HARDWARE SYSTEM SETUP	
2.0	BASIC CONCEPTS	27
	2.1 OVERVIEW	
	2.2 NUMBERING SYSTEMS	
	2.3 LADDER DIAGRAMS	
	2.4 COMMON CIRCUITRY	
3.0	MEMORY MATRIX AND ADDRESSING	61
	3.1 OVERVIEW	
	3.2 MEMORY ORGANIZATION AND ADDRESSING	
	3.3 MEMORY MATRIX	
4.0	PROGRAMMING	69
	4.1 OVERVIEW	
	4.2 IDENTIFYING PROGRAM REQUIREMENTS	
	4.3 START-UP PROCEDURE (MEMORY AND DATA CLEAR)	
	4.4 BASIC STATEMENT LOGIC	
	4.5 'AND LOAD' AND 'OR LOAD' STATEMENTS	
	4.6 INSERTIONS AND DELETIONS	
	4.7 ON/OFF-LINE OPERATIONS	
	4.8 ERRORS AND ERROR CODES	
	4.9 FUNCTION LOGIC	
	4.10 PROGRAM STORAGE	
	4.11 APPLICATION EXERCISES	

- This Page Intentionally Left Blank -

SECTION 1.0
INTRODUCTION

1.1 COURSE DESCRIPTION

The C20K Family Basic Applications course is designed to introduce the user to Omron's newest micro Programmable Controllers (PLCs). The course will begin with a description of system components which comprise the C20K. The user will become acquainted with the various numbering systems commonly used with PLCs as well as learn how to create Ladder Logic diagrams using the principles of Logic. After acquiring an understanding of memory organization and addressing, the trainee will obtain hands-on programming experience using the C20K controller.

1.2 INTRODUCTION TO PROGRAMMABLE CONTROLLERS

Called by many as the "industrial revolution of the '70s", the Programmable Controller (PLC) is a solid state device which houses electrical components including standard relays, latching relays, and electronic timers and counters normally used in electrical circuits. These components may be programmed or connected to one another to create virtually any type of electrical control circuit imaginable. The National Electrical Manufacturers Association (NEMA) defines a PLC as: "A digital electronic device with a programmable memory for storing instructions to implement specific functions such as logic, sequencing, timing, counting, and arithmetic to control machines and processes."

The PLC may be likened to a small computer designed to replace hard wired electromagnetic relay circuits. However, because PLCs are event-driven meaning that they respond to changes in the present status of a particular process, system, or device, they differ from general purpose computers which are driven largely by stored information. Table 1.1 provides a list of industries and applications where PLCs have been successfully utilized.

Table 1.1 Programmable Controller Industry Applications

INDUSTRY	APPLICATION
POWER	1. Fuel Handling 2. Burner Control 3. Fuel Control
PETROLEUM	1. Drilling 2. Pipeline 3. Processing
METALS	1. Heat Treatment 2. Ore Handling 3. Pelletizing 4. Fabrication
MINING	1. Ore Handling 2. Ore Loading 3. Ore Processing 4. Waste Treatment 5. Well (Water) Supply
FOOD/BEVERAGE	1. Process Control 2. Material Handling 3. Palletizing 4. Weighing 5. Canning 6. Bottling 7. Packaging & Filling
INVENTORY CONTROL	1. Order Pickers 2. Rack Selection 3. AGVS Monitoring
CHEMICAL/ PETROCHEMICAL	1. Waste Treatment 2. Water Treatment 3. Material Handling 4. Batch Control 5. Weighing 6. Blending
PAPER/PULP	1. Batch Digester 2. Water Treatment 3. Waste Treatment 4. Material Handling
MACHINE/TOOL	1. Lathe Control 2. Milling Machine 3. Grinders 4. Shapers
FLEXIBLE MANUFACTURING	1. Robotics 2. Inventory Control 3. Welding 4. Painting

1.3 C20K FAMILY SYSTEM COMPONENTS

The C20K Family of micro PLCs offers the most flexible I/O options as well as the widest variety of expansion units (modules). The C20K Family consists of the C20K, C28K, and C40K Central Processing Units (CPUs) equipped with 20, 28, and 40 I/O respectively. Each system may be expanded up to 140 I/O using expansion modules available in 4, 16, 20, 28, and 40 I/O point units. Reversible and high-speed drum sequencing control functions are standard. An optional timer module with four (4) analog-set timers is available. The C20K Family offers host computer and I/O link networking capabilities with other C-Series controllers and uses C-Series peripherals.

1.3.1 Central Processing Unit (CPU)

The Central Processing Unit (CPU) performs the decision-making arithmetic and program sequencing operations. The CPU must accommodate the memory and processing functions. Memory

includes the storage of data and programmed instructions while processing involves scanning and execution of the programmed user instructions. Located inside the CPU are a circuit board with memory chips, a microprocessor chip, and the necessary circuits for intercommunication.

Powered by an internal power supply (100 to 240 VAC or 24 VDC), the base CPU unit is available in 20, 28, and 40 I/O points and may be expanded up to 140 I/O (see Paragraph 1.3.2). The power supply is a device which converts an AC or DC voltage of specific value to one or more DC voltages of a specified value and current capacity. All PLC control systems require singular or multiple power sources to properly operate the internal logic and associated peripheral devices. Power supplies used with PLCs are designed to convert 120 or 240 VAC to DC to operate the CPU and I/O hardware.

The PLC must be able to store and retrieve information at will. To accomplish this, the PLC needs a memory bank in its CPU. The memory bank inside the C20K CPU contains a Random Access Memory (RAM) chip. In this type of memory, data may be retrieved (read) and stored (written) randomly. Data may be programmed word by word and individual words can be inserted, changed, or deleted without erasing the entire memory. The entire memory or individual parts of it, can however be erased and reprogrammed any number of times, if required. Because RAM systems are volatile, meaning they lose data with power loss, it is highly recommended that when a program is completed, it be saved onto a Read-Only Memory (ROM) chip for protection (see Section 4.10.2). The memory capacity is 1,194 addresses for each type of chip.

For a comprehensive list of CPU characteristics, refer to Table 1.2.

Table 1.2 CPU Characteristics

Main control	MPU, C-MOS, LS-TTL
Programming method	Ladder diagram
Instruction length	1 address/instruction, 6 bytes/instruction
Number of instructions	49
Execution time	10 μs/instruction (average)
Memory capacity	1,194 addresses
Internal auxiliary relays	136 (1000 to 1807) 1804 to 1806 are reserved for RDM: FUN 60, if it is used. 1807 is reserved as soft reset input for HDM: FUN 61, if it is used.
Special auxiliary relays	16 (1808 to 1907) Normally-ON, normally-OFF, battery failure, initial scan ON, 0.1-s pulse, 0.2-s pulse, 1.0-s pulse, etc.
Holding (retentive) relays	160 (HR 000 to 915)
Temporary memory relays	8 (TR0 to 7)
Data memory channels	64 (DM CH 00 to 63) DM CH 00 to 31 are reserved as upper and lower limit setting areas for RDM: FUN 60, if it is used. DM CH 32 to 63 are reserved as upper and lower limit setting areas for HDM: FUN 61, if it is used.
Timers/counters	48 (total of TIMs, CNTs, and CNTRs) TIM 00 to 47 (0 to 999.9 s) TIMH 00 to 47 (0 to 99.99 s) CNT 00 to 47 (0 to 9999 counts) CNTR 00 to 47 (0 to 9999 counts) CNT 46 serves as RDM: FUN 60. Likewise, CNT 47 serves as HDM: FUN 61. When these instructions are not used, CNT 46 and 47 can be used otherwise.
High-speed counter	Count input: 0000 Hard reset input: 0001 Soft reset: 1807 Max. response frequency: 2 kHz Preset count range: 0000 to 9999 No. of outputs: 16
Reversible drum counter	Reset: 1804 Count input: 1805 Reverse input: 1806 Preset count range: 0000 to 9999 No. of outputs: 16
Memory protection	Status of holding relays, present value of counters, and contents of data memory are retained during power failure.
Battery life	5 years at 25°C Battery life is shortened at temperatures higher than 25°C. Replace battery with new one within 1 week when ALARM indicator blinks.
Self-diagnostic functions	CPU failure (watchdog timer) Memory failure I/O bus failure Battery failure, etc.
Program check	Program check (executed on start of RUN operation) END instruction missing JMP-JME error Coil duplication Circuit error DIFU/DIFD over error IL/ILC error

1.3.2 Input/Output Expansion Modules

An Input/Output (I/O) device or module is a piece of equipment that exchanges signals with a PLC. Specifically, input devices send signals to the PLC while output devices receive signals from the PLC.

Input devices are designed to sense a particular condition in the environment including temperature, mechanical motion, pressure, and switch position. The user must determine what

types of inputs are needed for a particular control application. Typical input devices to choose include pushbuttons, limit switches, thumbwheels, selector switches, and photoelectric sensors.

Output devices respond to the signals they receive and change some aspect of their environment. Typical output devices the user may choose include motor starters, solenoids, valve actuators, indicator lights, and simple solid-state relays.

As stated in the preceding paragraphs, the C20K family of controllers may be expanded up to 140 I/O by using the 4, 16, 20, 28, and 40 point expansion modules (see Table 1.3).

Table 1.3 Family I/O Capabilities

DESCRIPTION	INPUTS	OUTPUTS
C20K CPU C28K CPU C40K CPU	12 16 24	8 12 16
20-point Expansion 28-point Expansion 40-point Expansion	12 16 24	8 12 16
16-point Expansion 4-point Expansion	16 inputs or 16 outputs 4 inputs or 4 outputs	

1.3.3 Communication Units

The C20K family of PLCs has been designed for maximum flexibility in expansion and networking capabilities. The optional I/O Link Unit allows the C20K family of controllers to report their I/O status to larger C-Series controllers such as the C120, C200H, C500, C1000H, and C2000H. The larger PLC must however have a Remote I/O Master Module to accept the I/O link unit.

Host Computer Link Modules allow C20K controllers to communicate directly to factory floor computers as well as personal and mini-computers.

1.3.4 Programming Console

The CPU receives its programming instructions electronically from a programming device. Omron calls its programming device the Programming Console (see Figure 1.1) and it allows the user to create new control programs, or edit and verify existing ones.

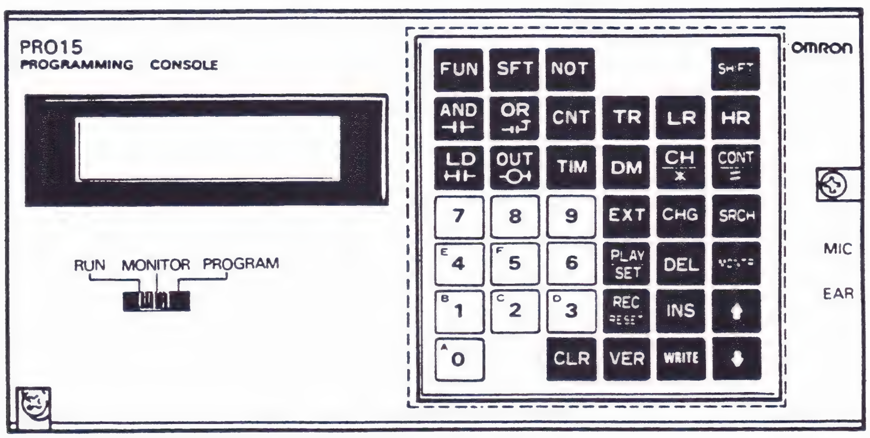


Figure 1.1 Programming Console

The keyboard of the Programming Console is functionally divided, by key color, into the following four areas:

White	Ten keys used to input program addresses, timing values, and other types of numeric entry.
Red	The CLEAR key.
Yellow	Twelve keys used to provide editing functions while writing and correcting the program.
Gray	Sixteen keys used to input instructions used in the program.

* NOTE *

Of the 16 gray keys, the LR key cannot be used with the K-type PCs.

A brief description of the yellow operation and gray instruction keys follows below:

Yellow Operation Keys

(a) EXT - The External key is used to interface with peripheral equipment.

(b) CHG - The Change key is used to change internal data such as counter, contact, and channel data.

(c) SRCH - The Search key is used to locate a specific address or function.

(d) PLAY/SET

PLAY - When the PLAY/SET key is pressed after the SHIFT key, a program may be uploaded from a cassette recorder.

SET - When the PLAY/SET key is pressed alone, a Set function is performed on the status of internal relays, counters, and timers.

(e) DEL - The Delete key is used to delete an instruction.

(f) MONTR - The Monitor key is used to monitor desired functions such as the status of internal relays, counters, timers, etc.

(g) REC/RESET

REC - When the REC/RESET key is pressed after the SHIFT key, a program may be downloaded to a cassette recorder.

RESET - When the REC/RESET key is pressed alone, a Reset function is performed on the status of internal relays, counters, and timers.

(h) INS - The Insert key is used to insert an instruction in a program.

(i) UP ARROW - The Up Arrow key is used to step up or decrement one program address at a time.

(j) VER - The Verify key is used to verify the contents of the I/O table.

(k) WRITE - The Write key is used to enter an instruction into the actual memory of the PLC.

(l) DOWN ARROW - The Down Arrow key is used to step down or increment one program address at a time.

<u>Gray Instruction Keys</u>

Except for the SHIFT key in the upper right hand corner of the console, the gray keys are used to enter instructions into a program. The SHIFT key is similar to a shift key on a typewriter. It enables the operator to use the second function of a key that has two functions.

A brief description of the remaining gray instruction keys is provided below:

(a) FUN - The Function key is used to select special functions. This is accomplished by pressing FUN and a numerical value.

(b) SFT - The Shift Register key is used to enter a Shift Register function.

(c) NOT - The NOT key is used to form a Normally Closed contact.

(d) AND - The AND key is used to enter an AND instruction for "ANDing" two contacts.

(e) OR - The OR key is used to enter an OR instruction for "ORing" two contacts.

(f) CNT - The Counter key is used to enter a Counter instruction and must be followed by counter data.

(g) LD - The Load key is used to enter a Load instruction for a specified input.

(h) OUT - The Output key is used to enter an output instruction for a specified output.

(i) TIM - The Timer key is used to enter a Timer instruction and must be followed by timer data.

(j) TR - The Temporary Relay key is used to enter a Temporary Memory Relay instruction.

(k) LR - The Link Relay key is used to address the Link Relay area.

(l) HR - The Holding Relay key is used to address the Holding Relay area.

(m) DM - The Data Memory key is used to address the Data Memory area.

(n) CH/*

CH - When the Channel key (CH/*) is pressed after the SHIFT key, a channel may be specified.

* - The CH/* key is also used with the DM key to call out an indirect address data memory (not available on K-Family PLCs).

(o) CONT/#

CONT - When the Contact key (CONT/#) is pressed after the SHIFT key, it may be used to search for a contact.

- When the CONT/# key is pressed alone, it is used to call out a numerical value ranging from 0000 to FFFF.

A three-position slide switch enables the user to select one of the three operating modes: RUN, MONITOR, or PROGRAM. An LCD display (located in the upper left hand corner of the unit) shows the current mode selected and also displays the input instructions as they are entered.

(1) The RUN mode is used to begin PLC operation. In this mode, the PLC controls the equipment using the program that has been entered into the PLC memory. The RUN mode also allows limited access to certain programming functions.

(2) The MONITOR mode may also be used to begin PLC operation. It enables the operator to visually monitor the operation in progress. In this mode, the status of

a particular relay may be checked by moving to its address.

(3) The PROGRAM mode is used during the actual programming of the PLC. A program cannot be entered in any mode other than PROGRAM.

1.4 HARDWARE SYSTEM SETUP

The following pages will concentrate on assembling a fully working control system. Understandably, each user has different system needs in terms of I/O configuration (including power requirements), I/O size, and physical size and placement of the system.

The previous paragraphs discussed the various hardware needed to make a working system including:

(1) CPU
(2) Memory (user)
(3) I/O
(4) Programming device (for writing the user program)

Now, the focus will be on assembling the various hardware and the different configurations available to the user.

1.4.1 C20K Family CPUs

As stated previously, the 'K' Family of C-Series controllers is available with different I/O configuration sizes on the CPU base unit:

(a) C20K - 12 inputs/8 outputs
(b) C28K - 16 inputs/12 outputs
(c) C40K - 24 inputs/16 outputs

Unlike other members of the C-Series controllers, the I/O on the CPU rack is dedicated. In the case of the C20K CPU, I/O numbers 0000-0011 will always be inputs and 0100-0107 will always be outputs. I/O connected to the CPU rack either directly or via an I/O connecting cable are known as "Local I/O". A more in-depth discussion of the I/O concept will follow later in this section.

Different power configurations and different local I/O signal levels are available with the C20K Family CPUs. For example, the C20K-CDR-A unit is powered by 100-240 VAC and contains twelve 24 VDC inputs and eight (8) socket-mounted relay outputs. Twelve inputs plus eight outputs equals 20 total CPU I/O.

The C40K-CDR-D on the other hand, is powered by a 24VDC power source and contains twenty four 24VDC input points and sixteen (16) socket-mounted relay output points. Twenty four points plus 16 points equals 40 total Local I/O.

Before examining the physical architecture of the K-Family PLCs, it is important to understand the numbering system used to call out different configurations of the CPU (see Figure 1.2)

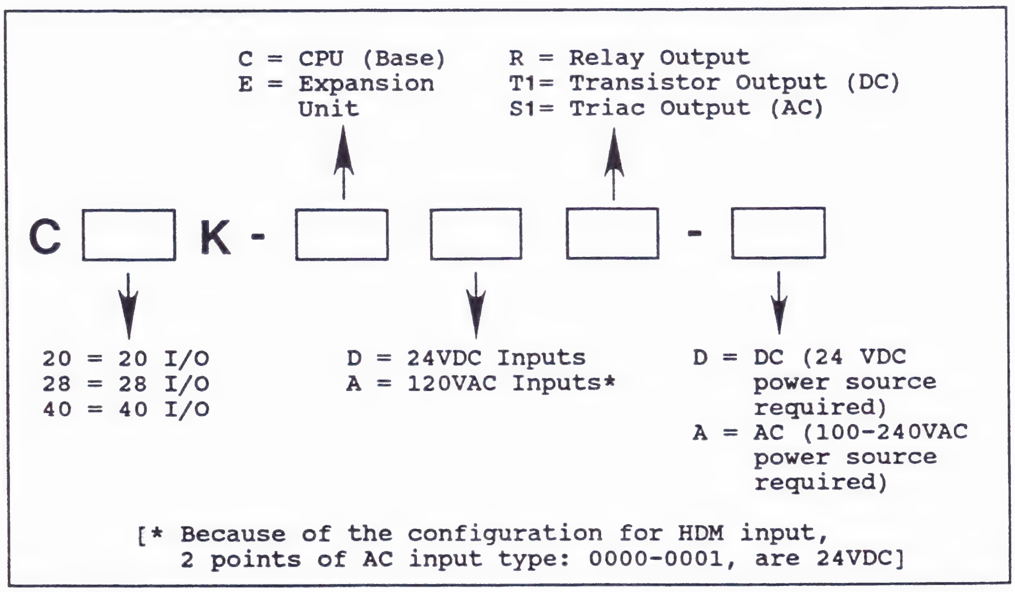


Figure 1.2 Part Number Determination

1.4.1.1 Physical Architecture of a CPU

To illustrate the physical characteristics and describe the nomenclature of the hardware, the C20K-CDR-A unit will serve as our example. Refer to Figure 1.3 for a physical layout of the C20K CPU.

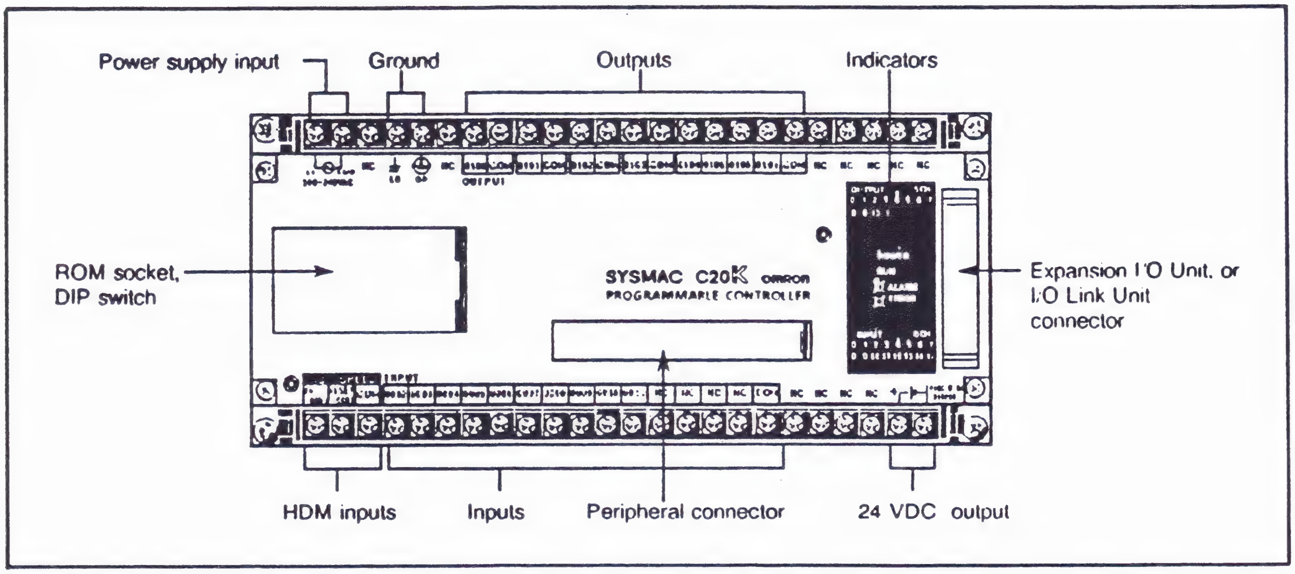


Figure 1.3 C20K Programmable Controller

(a) Power Supply Input - Depending upon which CPU is chosen, you will have an input for:

AC - 100 to 240 VAC (operating range of 85-264 VAC) with a power consumption of 60 VA maximum.

DC - 24 VDC (operating range of 20.4-26.4 VDC) with a power consumption of 40 watts maximum.

(b) Ground terminals -

GR: Ground terminal (electric shock prevention).
LG: Line ground (noise filter; neutral).

(c) 24 VDC Output - Built-in power supply that may be used to power outboard devices (photoelectrics, proximity sensors, etc.). The output current of this power supply is limited to 0.3 amps (300 millimeters). If more than 0.3 amps are required to drive your device, use a separate power supply.

(d) Expansion I/O Connection Point - See Paragraph 1.3.2 for an explanation.

(e) Peripheral Connector - For connecting devices such as Programming Console, GPC, P-ROM writer, Printer Interface unit.

(f) Inputs - See Paragraph 1.4.2.1.

(g) Outputs - See Paragraph 1.4.2.2.

(h) HDM Inputs - See Paragraph 1.4.2.

(i) ROM Socket and DIP Switch - Each of the C-Series K-Family controllers is provided with a built-in RAM (backed up by a Lithium battery), as well as a ROM chip socket. The memory capacity is 1194 addresses for each. To install an EPROM chip refer to Figure 1.4 and follow the steps below.

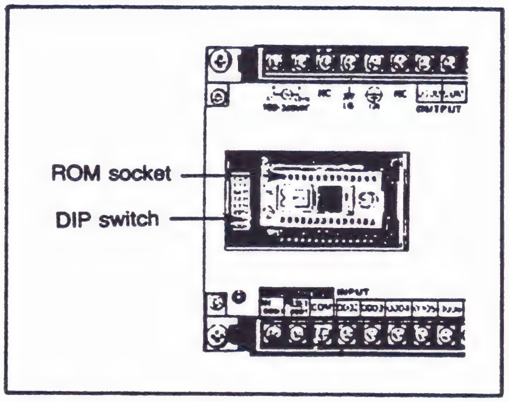


Figure 1.4 ROM Socket and Dip Switch

1. Remove the cover from the CPU (use a screwdriver if necessary).
2. Raise the lever to unlock the socket.
3. Holding the chip so that you do not touch the pins, insert it into the socket with the notch directed as shown.
4. Lower the lever to lock the chip in.
5. Replace the cover on the CPU.

DIP Switch Setting

Note that only DIP switch pins 1 and 2 are ON when the PLC is shipped from the factory.

Listed below are DIP switch settings for all C-Series K-Family controllers.

8 7	Turn ON to use hard reset (0001) Turn OFF if FUN 61 is not used.
6	Turn ON for English display
5	Turn ON to inhibit ALARM blinker.*
4 3	ROM: ON (RAM: OFF)
2 1	RAM: ON (ROM: OFF)

Dip Switches:
- 1,2 - CPU processes program in RAM memory
- 3,4 - CPU processes program in ROM memory
- 5 - Alarm indication for "LOW" battery
- 6 - English display
- 7,8 - Used when HDM is addressed in a program

* NOTE *

> Battery backs up user memory, as well as Data Memory, Holding Relay area, etc.

1.4.2 I/O Assignments

As discussed earlier, the number of I/O points depends upon which CPU (C20K, 28K, 40K) is used.

Figure 1.5 shows the exact wiring of the I/O terminals and common terminals, as well as the power supply connections. A C40K CPU is used as an example, but the other CPUs have the same characteristics, except that they have fewer terminals. The 'NC' terminals are not connected internally.

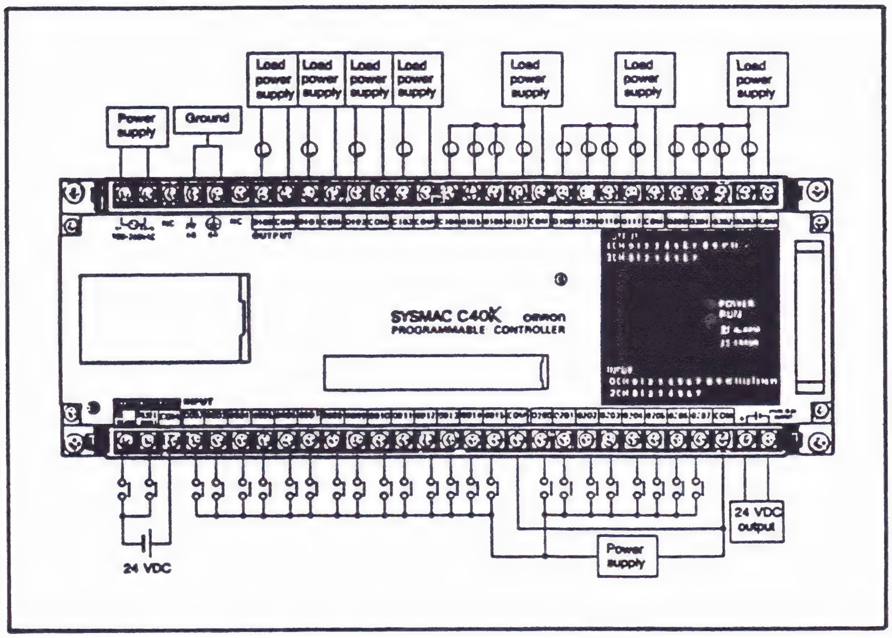


Figure 1.5 Wiring Connections for a C40K

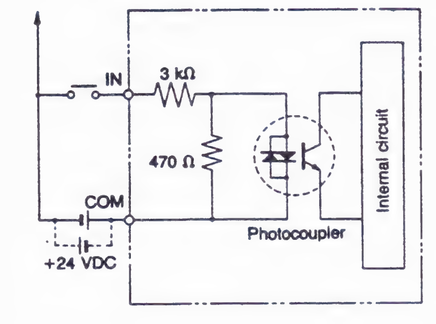
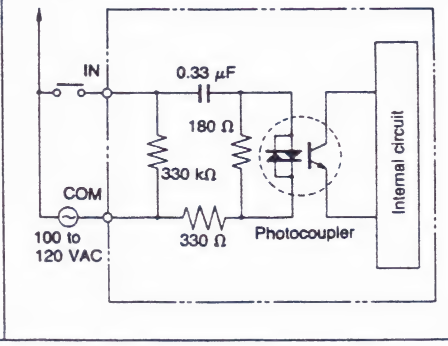
The connections for the Expansion I/O Units are identical except that the first two inputs of the CPU (0000 and 0001) are for the High-speed Counter (HDM), and the first two inputs of Expansion I/O Units are for general use.

If HDM is not used, Inputs 0000 and 0001 may be used for general use. However, the response time is shorter than the other inputs (0.15 ms).

1.4.2.1 Input Specifications

Table 1.4 contains specifications for the standard discreet inputs of the C20K Family. Both DC and AC input specifications are covered.

Table 1.4 Input Specifications

	DC input (photocoupler-isolated)	AC input * (photocoupler-isolated)
Supply voltage	24 VDC±10%	100 to 120 VAC+10%, −15% 50/60 Hz
Input impedance	3 kΩ	9.7 kΩ (50 Hz), 8 kΩ (60 Hz)
ON voltage	7 mA at 24 VDC	10 mA at 100 VAC
ON voltage	15 VDC min.	60 VAC min.
OFF voltage	5 VDC max.	20 VAC max.
ON delay time	2.5 ms max. (input 0000 and 0001: 0.15 ms)	35 ms max.
OFF delay time	2.5 ms max. (input 0000 and 0001: 0.15 ms)	55 ms max.
Circuit configuration		

* NOTES *

(1) All internal logic circuits are powered by the PLC.

(2) Delay time is the time that signal level (voltage) must be present/absent before the PLC will recognize the condition of a signal.

(3) Input points per common varies per I/O configuration of a system. Consult the system manual for this information.

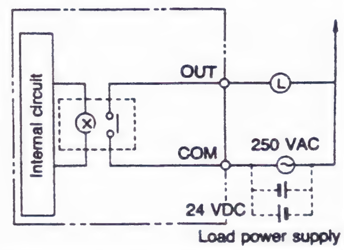
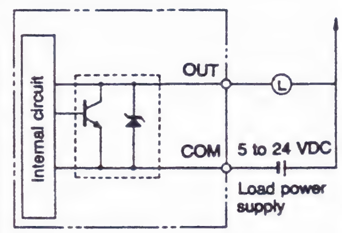
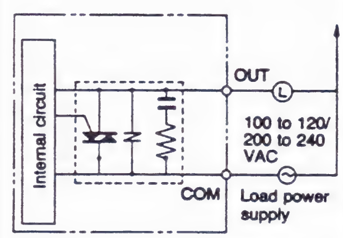
(4) Inputs 0000 and 0001 operate on DC input voltage. The circuit configuration of these two points is the same as the DC input circuit shown above.

(5) The 24 VDC power source can be connected to either the positive or the negative terminal. Therefore, both the PNP input (negative common) and NPN input (positive common) can be used.

1.4.2.2 Output Specifications

Table 1.5 contains the output specifications for the C20K Family.

Table 1.5 Output Specifications

	ON-delay	OFF-delay	Switching capacity Max.	Switching capacity Min.	Circuit configuration
Relay (photocoupler-isolated)	15 ms max.	15 ms max.	2 A at 250 VAC 2 A at 24 VDC ($\cos\varphi$ = 1) 0.5 A at 250 VAC ($\cos\varphi$ = 0.4) A/4 points	10 mA at 5 VDC	
Transistor* (photocoupler-isolated)	1.5 ms max.	1.5 ms max.	1 A/point at 5 to 24 VDC, 1.6 to 4 A/4 points	10 mA at 5 VDC, saturation voltage: 1.5 V max.	
Triac** (photocoupler-isolated)	1.5 ms max.	1/2 of load frequency + 1 ms max.	1 A/point at 85 to 250 VAC, 1.6 to 4 A/4 points	10 mA at 100 VAC, 20 mA at 200 VAC	

* Leakage current: 100 μA max. at 24 VDC
Residual voltage: 1.5 V max.
** Leakage current: 20 mA max. at 100 VAC/5 mA max. at 200 VAC
Residual voltage: 1.5 V max.

* NOTES *

(1)	All internal logic circuits are powered by the PLC.
(2)	Output points per common varies according to the I/O configuration of a system. Consult the appropriate system manual for this information.
(3)	Triac incorporates "zero-crossing" network.

1.4.3 Expansion I/O (Remote I/O)

The system I/O capability can be expanded to include a greater number of points than are resident on the CPU. Omron offers three styles of remote I/O expansion units with different I/O point capabilities. The three expansion units come in 20-point, 28-point, and 40-point configurations (see Figure 1.6). All remote I/O follows the same numbering scheme as the CPU types (refer back to Figure 1.2) except that the 'K' designation changes to 'P' to indicate expansion I/O. For example, C20P-EDR-A is an expansion unit which contains 12 input points powered by a 100 VAC power source, with input voltage of 24 VDC, and 8 relay output points.

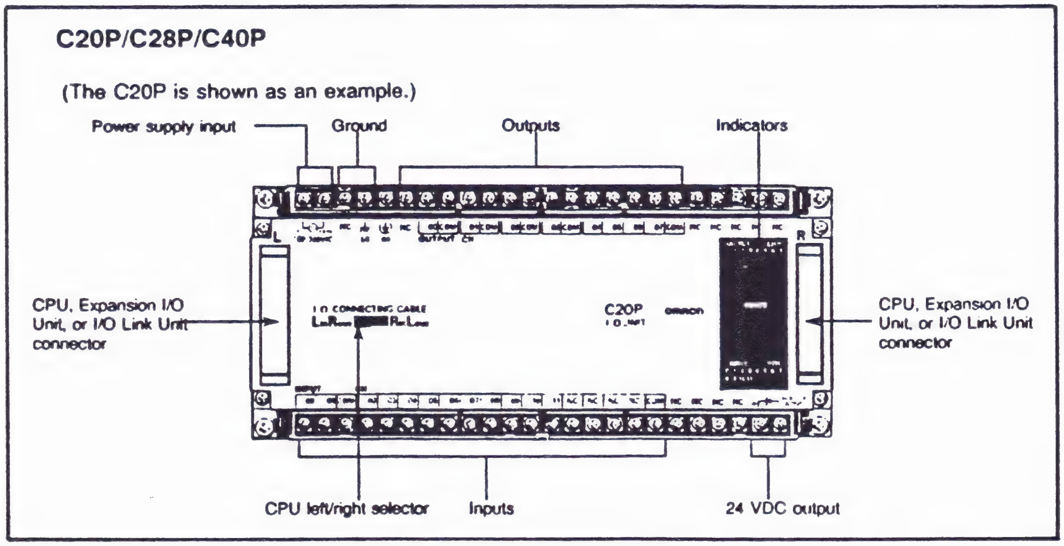


Figure 1.6 Expansion I/O Unit (C20P)

The expansion I/O units are available in the same I/O density and I/O types as the CPU I/O.

The figures below depict the physical location of the expansion connection ports in relation to the CPU and expansion units. If the CPU Left/Right selector switch on the Expansion I/O units is placed in the left position (L IN; R OUT) this signifies "Left IN" from the CPU and "Right OUT" to further expansion units (see Figure 1.7).

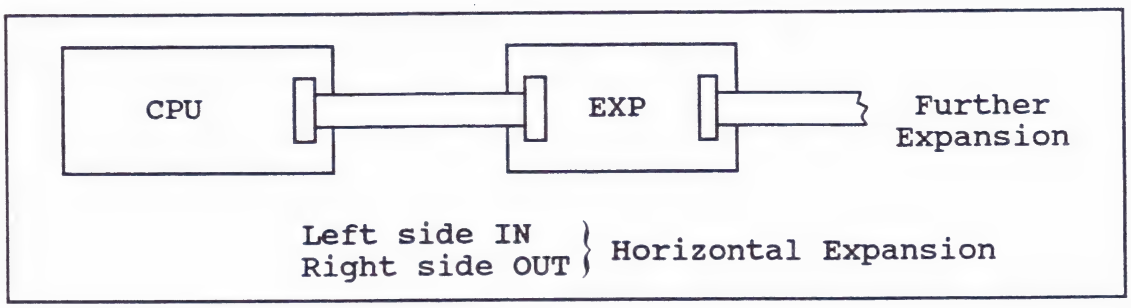


Figure 1.7 Horizontal Expansion (Left IN, Right OUT)

If the selection switch is placed in the right position (R IN, L OUT) this signifies a "Right IN" from the CPU and a "Left OUT" to further expansion units (Figure 1.8).

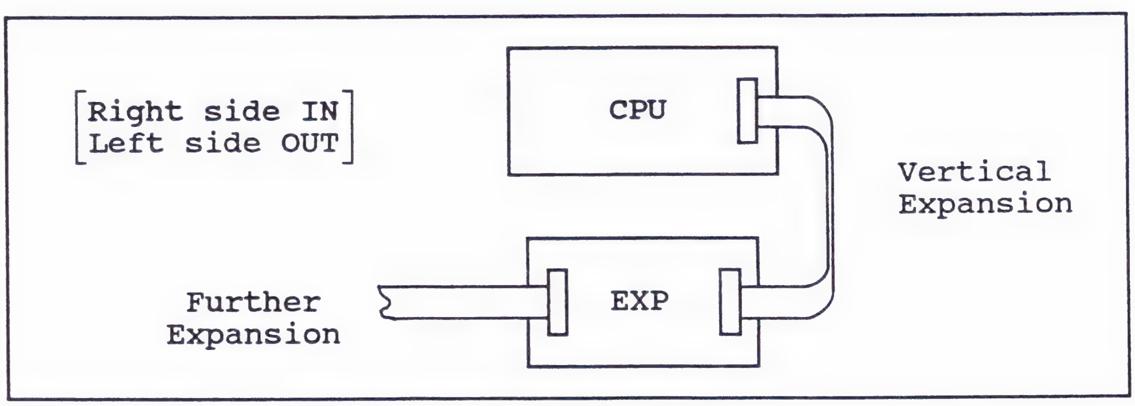


Figure 1.8 Vertical Expansion (Right IN, Left OUT)

In addition to the 20-, 28-, and 40-point expansion units, Omron offers 16- and 4-point I/O expansion units as shown in Figure 1.9.

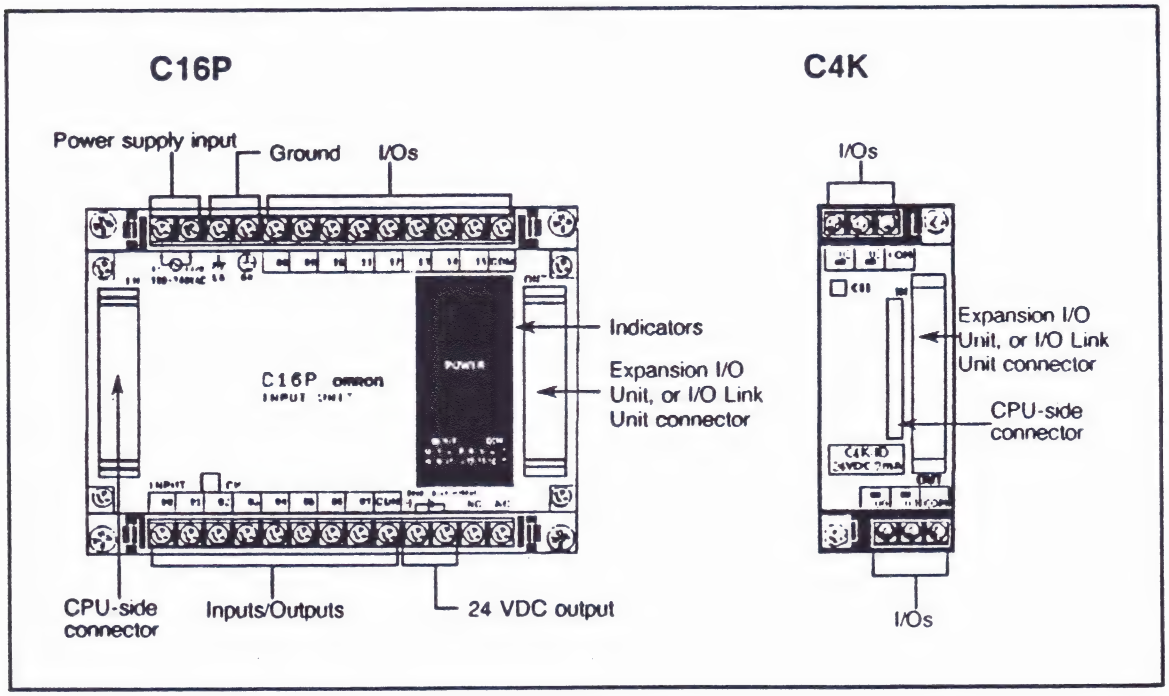


Figure 1.9 16- and 4-Point Expansion Units

The C16P unit is available with 16 input points or 16 output points. Note that the connection side is fixed. The C4K unit contains 4 input points or 4 output points (also fixed).

1.4.3.1 Determining I/O Capability

The C20K Family can address five separate address areas in the total system. In other words, the number of "parts" to a system is limited to a maximum of five, and as a result I/O capacity will vary.

C20K, C28K C20P, C28P C16P, C4k	1 part each
C40K, C40P	2 parts each

For example, a control system that consists of:

(a) C20K + C20P + C40P + C4K (5 parts)
= 84 I/O, the maximum for this system.

(b) C28K + C20P + C28P + C28P + C4K (5 parts)
= 108 I/O, the maximum for this system.

(c) C20K + four C4Ks (5 parts)
= 36 total addressable I/O.

1.4.4 I/O Addressing

Addressing for the C20K Family is similar to all Omron C-Series Programmable Controllers in that channel locations and channel point make up the address structure. For example, 0102 would designate Channel 01, Point 02.

An in-depth discussion on addressing and memory matrixes is found in Section 3.0, however, a brief introduction is provided below using two examples. The first example outlines the addressing scheme of a C20K CPU plus expansion I/O, and the second example uses a C40K CPU plus expansion I/O.

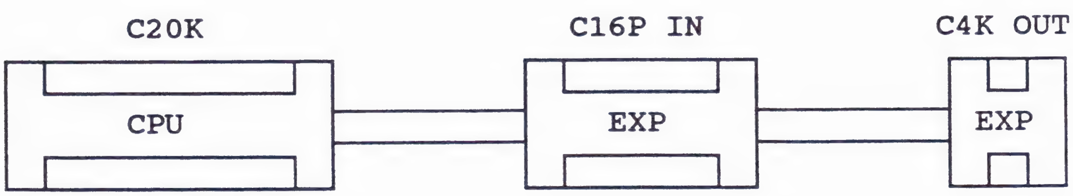
Table 1.4 shows the addressing for both CPUs.

Table 1.4 Addressing For C20K And C40K CPUs

CPU	CHANNEL #	INPUTS	OUTPUTS	REAL I/O ADDRESS RANGE
C20K	CHANNEL 00	12 PTS.		0000 - 0011*
	CHANNEL 01	–	8 PTS.	0100 - 0107*
C40K	CHANNEL 00	16 PTS.		0000 - 0015
	CHANNEL 01	–	12 PTS.	0100 - 0111
	CHANNEL 02	8 PTS.	–	0200 - 0207 **
	CHANNEL 03	AUX. RELAYS		0300 - 0303

*0012 - 0015 AND 0108 - 0115 MAY BE USED AS AUXILIARY RELAYS.
** 0209 - 0215 MAY BE USED AS AUXILIARY RELAYS.

<u>Example No. 1</u>: C20K (Local) plus Expansion I/O

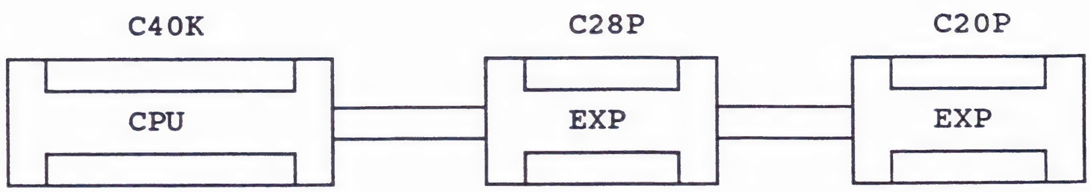


CPU / EXPANSION RACK	I/O CHANNEL DESIGNATION	ADDRESS RANGE	AUXILIARY RELAYS
C20K (CPU)	00 - INPUT 01 - OUTPUT	0000 - 0011 0100 - 0107	0012 - 0015 0108 - 0115
C16P (EXP. RACK)	02 - INPUT	0200 - 0215	NONE
C4K (EXP. RACK)	03 - OUTPUT	0300 - 0303	0304 - 0315

Thus, the total system I/O adds up to 40:

28 inputs
12 outputs

Example No. 2: C40K (Local) plus Expansion I/O



CPU / EXPANSION RACK	I/O CHANNEL DESIGNATION	ADDRESS RANGE	AUXILIARY RELAYS
C20K (CPU)	00 - INPUT 01 - OUTPUT 02 - INPUT 03 - OUTPUT	0000 - 0015 0100 - 0111 0200 - 0207 0300 - 0303	NONE 0112 - 0115 0208 - 0215 0304 - 0315
C28P (EXP. RACK)	04 - INPUT 05 - OUTPUT	0400 - 0415 0500 - 0511	NONE 0512 - 0515
C20P(EXP. RACK)	06 - INPUT 07 - OUTPUT	0600 - 0611 0700 - 0707	0612 - 0615 0708 - 0715

Total system I/O adds up to 88:

52 inputs
36 outputs

- This Page Intentionally Left Blank -

SECTION 2.0
BASIC CONCEPTS

2.1 OVERVIEW

This section is designed to introduce the user to the numbering systems commonly used to code and transmit data in a PLC. The concepts and symbols used in Ladder Logic diagrams will be covered and in addition, an introduction to the principles of Logic will be provided.

2.2 NUMBERING SYSTEMS

Because PLCs are digital devices, they respond to only two states (conditions): ON and OFF. The numerical digit "1" represents the ON state, and the numerical digit "0" represents the OFF condition. Because of this limited amount of conditional statements, "1s" and "0s" must be grouped together in order to represent more than one state or condition.

Five numbering systems are commonly used with PLCs, including Decimal, Binary, Octal, Hexadecimal, and Binary-coded Decimal. The following paragraphs will describe all but the Octal numbering system.

2.2.1 Decimal

The Decimal numbering system is familiar to everyone. In the Decimal system there are only 10 digits used to represent any given quantity. These digits are represented by the numerals 0,1,2,3,4,5,6,7,8, and 9. There may be more esoteric or mysterious ways in which the Decimal system orginated, but the main reason is because we have ten fingers. These ten fingers may represent ten different states or conditions. Since the Decimal system has ten discrete digits, it has a "base" value of ten (10). To identify a number using the Decimal numbering system, each digit (beginning from the right-most to the left-most) is interpreted in ascending powers of ten. For example, the number 106,324 consists of one hundred and six thousand, three hundred and twenty four units. By breaking this number up into powers of 10 we get the following:

	10^5	10^4	10^3	10^2	10^1	10^0
Base Location	10^5	10^4	10^3	10^2	10^1	10^0
Numerical Value	1	0	6	3	2	4

10^0 = 1s
10^1 = 10s
10^2 = 100s
10^3 = 1,000s
10^4 = 10,000s
10^5 = 100,000s

So, each digit represents a numerical value times (X) its base location as follows:

$$\begin{aligned}
1 \times 10^5 &= 100{,}000 \\
0 \times 10^4 &= 0 \\
6 \times 10^3 &= 6{,}000 \\
3 \times 10^2 &= 300 \\
2 \times 10^1 &= 20 \\
4 \times 10^0 &= \underline{\quad 4}
\end{aligned}$$

* (Remember a number raised to "0" power is equal to 1.)

Add these up: 106,324

2.2.2 Binary

The Binary numbering system, by contrast, uses only two digits, zero (0) and one (1) and has a base value of two (2). PLCs utilize the Binary system because they achieve only one of two stable states (conditions), ON or OFF (1 or 0). Thus by assigning the value '1' to a working condition and the value '0' to an inactive state, a circuit may be turned ON or OFF, high or low, or perform any opposite pair of operations. Also, by giving a weighted numerical value to a group of ON/OFF signals, mathematical notation is used to solve control problems.

10011001 is an example of a Binary number representing a certain status. There are 8 points (0-7) to consider. Again, since there are two status conditions, we use a base value of 2 to split the number as follows:

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary Digits	1	0	0	1	1	0	0	1

$$
\begin{aligned}
2^0 &= 1\\
2^1 &= 2\\
2^2 &= 4\\
2^3 &= 8\\
2^4 &= 16\\
2^5 &= 32\\
2^6 &= 64\\
2^7 &= 128
\end{aligned}
$$

For example, $2^5 = 2 \times 2 \times 2 \times 2 \times 2 = 32$, in the same manner as $10^3 = 10 \times 10 \times 10 = 1000$.

So, to complete our conversion:

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary Digits	1	0	0	1	1	0	0	1
Decimal Value	128	--	--	16	8	--	--	1

Thus, by adding 128 + 16 + 8 + 1, the result is 153. Therefore, 10011001 = 153.

Example No. 2

To convert the status value 10110 in descending order:

$$
\begin{aligned}
1 \times 2^4 &= 1 \times (2 \times 2 \times 2 \times 2) &= 16\\
0 \times 2^3 &= 0 \times (2 \times 2 \times 2) &= 0\\
1 \times 2^2 &= 1 \times (2 \times 2) &= 4\\
1 \times 2^1 &= 1 \times (2) &= 2\\
0 \times 2^0 &= 0 \times (1) &= \underline{0}\\
&\text{Add all the values} &= 22
\end{aligned}
$$

Two important terms are used to define value placement in binary numbers:

(a) Most Significant Bit (MSB)
(b) Least Significant Bit (LSB)

Note the following using the value used in the example above:

100110

MSB LSB

<u>Example No. 3</u>

Convert the status value 111111101:

Powers of 2	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary Digits	1	1	1	1	1	1	1	0	1

1×2^8	=1 x (2x2x2x2x2x2x2x2)	=	1 x 256	=	256
1×2^7	=1 x (2x2x2x2x2x2x2)	=	1 x 128	=	128
1×2^6	=1 x (2x2x2x2x2x2)	=	1 x 64	=	64
1×2^5	=1 x (2x2x2x2x2)	=	1 x 32	=	32
1×2^4	=1 x (2x2x2x2)	=	1 x 16	=	16
1×2^3	=1 x (2x2x2)	=	1 x 8	=	8
1×2^2	=1 x (2x2)	=	1 x 4	=	4
0×2^1	=0 x (2)	=	0 x 2	=	0
1×2^0	=1 x (1)	=	1 x 1	=	1

Add all value representations:

256 + 128 + 64 + 32 + 16 + 8 + 4 + 0 + 1 = 509

2.2.3 Binary Code Decimal

In all reality, Binary Coded Decimal (BCD) is not a numbering system, but rather a modification of the Binary system. Retaining the characteristics of both Binary and Decimal

numbering systems, BCD uses Binary digits to represent a Decimal value.

The ground rules for BCD are as follows:

(a) Four bits (status/conditions) equal one BCD digit.
(b) Each BCD digit represents a Base 10 value. For example,

$10^0 = 1s$, $10^1 = 10s$, $10^2 = 100s$, etc.

BCD is easier to understand when examining a large number. Since BCD requires 4 bits per digit, it is only necessary to take the base value of 2 to the third power as follows:

____	____	____	____
2^3	2^2	2^1	2^0
8	4	2	1

0100 0000 0100 is an example of a BCD value representing a certain status.

10^2	10^1	10^0
<u>0100</u>	<u>0000</u>	<u>0100</u>
8421	8421	8421

$$
\begin{aligned}
4 \times 10^0 &= 4 \\
0 \times 10^1 &= 0 \\
4 \times 10^2 &= \underline{400}
\end{aligned}
$$

Add these up to obtain decimal value (Base 10) = 404

Example No. 2

$$\begin{array}{|c|c|c|} \multicolumn{1}{c}{10^2} & \multicolumn{1}{c}{10^1} & \multicolumn{1}{c}{10^0} \\ \underline{1001} & \underline{0001} & \underline{0111} \\ 8421 & 8421 & 8421 \end{array}$$

$$\begin{array}{rcr} 7 \times 10^0 & = & 7 \\ 1 \times 10^1 & = & 10 \\ 9 \times 10^2 & = & \underline{900} \\ & & 917 \end{array}$$

Note that the sum of 4 bits representing a Base 10 (Decimal) value can never be greater than 9. Anything greater would be in the next base or multiplier.

Table 2.1 compares the Decimal, Binary, and BCD numbering systems.

DECIMAL DIGIT	BINARY DIGIT	BINARY CODED DECIMAL
0	0000	0000 0000
1	0001	0000 0001
2	0010	0000 0010
3	0011	0000 0011
4	0100	0000 0100
5	0101	0000 0101
6	0110	0000 0110
7	0111	0000 0111
8	1000	0000 1000
9	1001	0000 1001
10	1010	0001 0000

TABLE 2.1 BINARY CODED DECIMAL SYSTEM

2.2.4 Hexadecimal

The Hexadecimal numbering system is another compact numbering system used in programming that allows us to obtain greater values than BCD. The Hexadecimal system has a base of sixteen (16) represented by the decimal digits 0 through 9 and alphabetical characters A through F (A=10, F=15):

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

HEXADECIMAL	BINARY DIGIT	BINARY CODED DECIMAL
0	0000	0000 0000
1	0001	0000 0001
2	0010	0000 0010
3	0011	0000 0011
4	0100	0000 0100
5	0101	0000 0101
6	0110	0000 0110
7	0111	0000 0111
8	1000	0000 1000
9	1001	0000 1001
A	1010	0001 0000
B	1011	0001 0001
C	1100	0001 0010
D	1101	0001 0011
E	1110	0001 0100
F	1111	0001 0101

TABLE 2.2 HEXADECIMAL EQUIVALENTS

10F0 is an example of a Hexadecimal number. To convert this number to its decimal equivalent, multiply each digit by its Base 16 positional value and add the results:

1 0 F 0

$$\begin{aligned} 0 \times 16^0 &= 0 \\ 15 \times 16^1 &= 240 \\ 0 \times 16^2 &= 0 \\ 1 \times 16^3 &= \underline{4096} \\ & 4336 \end{aligned}$$

Because Omron controllers use 16-bit data (words), we can numerically monitor the status of all 16 bits.

16^3	16^2	16^1	16^0
1011	0111	1010	1100
8421	8421	8421	8421

This is called Binary Coded Hexadecimal.

For example, the bits under 16^0 through 16^3 are:

$$
\begin{aligned}
16^0 &= 1 + 4 + 8 = D \\
16^1 &= 2 + 8 = A \\
16^2 &= 1 + 2 + 4 = 7 \\
16^3 &= 1 + 2 + 8 = B
\end{aligned}
$$

So, the number displayed would be B7AD. To convert to Decimal:

B 7 A D

$$
\begin{aligned}
13 \times 16^0 &= 13 \\
10 \times 16^1 &= 160 \\
7 \times 16^2 &= 1792 \\
11 \times 16^3 &= \underline{45056} \\
 & 47021
\end{aligned}
$$

The Binary Coded Hexadecimal numbering system then, can be used to express a large value in only 4 digits.

2.2.5 BCD Thumbwheels

One of the best ways to illustrate BCD is by understanding how a BCD Thumbwheel operates. Figure 2.1 shows an example of a pushbutton BCD Thumbwheel.

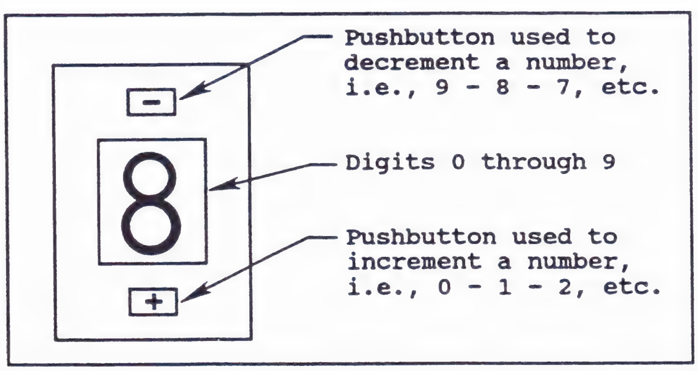
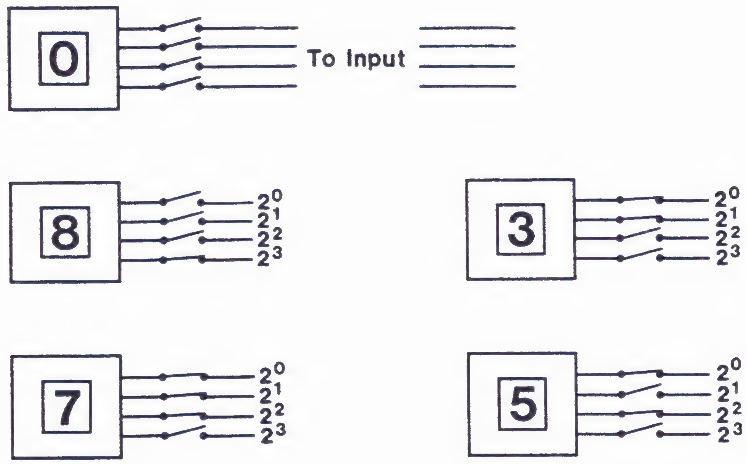


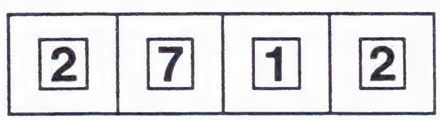
Figure 2.1 BCD Thumbwheel

See the following schematics to understand how certain numbers are obtained.



Remember, four points are used to identify numerical values 0 through 9.

When BCD thumbwheels are mounted together, they can represent more than one digit as shown below.



Remember, every digit requires 4 bits (points).

2.2.6 Exercises

Solve the following problems:

<u>Exercise No. 1</u>

Convert the Binary value 10010 to Decimal:

<u>Exercise No. 2</u>

Convert the Decimal value 372 to a Binary value:

<u>Exercise No. 3</u>

Convert the Decimal value 1000 to a BCD value:

Exercise No. 4

What is the largest value you can have using:

12-bit Binary?

16-bit BCD?

5-digit Decimal?

2.3 LADDER DIAGRAMS

Although PLC memory configuration (designs) vary somewhat from manufacturer to manufacturer, most user memories are set up to receive instructions in the same general way. Regardless of the type of programming device, control commands are usually entered into memory using Relay Ladder Logic.

An Electrician's Ladder Diagram is a matrix (network) of available power paths to one or more electrical outputs. The two vertical lines of a Ladder diagram represent the main power bus while the rungs contain a network of hard-wired contacts and relays. For PLC programming, the symbols in the traditional Electrician's Ladder diagram are slightly modified. Table 2.2 provides a comparison of Hard-wired Electrician's and PLC Ladder Logic diagrams. It is important to note that the symbols in the two types of Ladders mean vastly different things. For example, each contact and relay present in a Hard-wired Ladder correspond to locations in input or output registers in PLC programming. There are actually no relay coils present in a PLC. The relay symbol merely represents a bit in the output register -- a bit to be set ON or OFF during the execution of the user program.

Each rung of a PLC Ladder diagram corresponds to a set of instructions which tell the PLC what to do in response to the status of a given set of inputs (contacts). The contact is the most common symbol used in PLC programming. Two different symbols are used to represent contacts: one for Normally Open (NO) and a second for Normally Closed (NC). These contact symbols may represent inputs in the input register or outputs in the output register.

One way to understand the contacts in a user program is to think of them as commands to the PLC. A Normally Open contact symbol is a command to search for an ON or "working" condition while a Normally Closed contact symbol is a command to search for an OFF or "non-working" condition. A "working" or "true" condition is granted at the contact if the PLC finds the required bit condition. If the PLC finds a "true" condition for every contact on the run, the output bit is set to either ON or OFF as the output symbol dictates. A summary of the ON and OFF commands is given in Table 2.3.

Table 2.2 Ladder Diagram Comparison

	ELECTRICIAN'S HARD-WIRED LADDER	PLC LADDER LOGIC
Reason for Using a Ladder Diagram	Shows wiring plan for connecting components	Aid to Programming
Symbols:		
Vertical Lines	Main Bus	Beginning and end of rungs
Rungs	Branch Circuits	Instruction Sets
-| |- -|/|-	Contacts	Addresses for input and output devices
-( )- -(/)-	Electromagnetic Relays	Addresses in output register
Implementation	Connect wires following schematic	Key-in symbols on programming device

Table 2.3 Normally Open (NO) and Normally Closed (NC) Contacts

Contact Type	Symbol	Commands Given to PLC	PLC Establishes Continuity if Searched-for Bit is:
NO	-| |-	Search for an ON Condition	1 (ON)
NC	-|/|-	Search for an OFF Condition	0 (OFF)

When implementing these commands, it is important to determine whether the symbol in the program represents an input or an output. Figure 2.2 contains a detailed list of input and output symbols depicting both Electrician's Ladder symbols and their Ladder Logic equivalents. A Normally Open input contact tells the PLC to scan the input register for its designated input location and check it for an ON condition. If the PLC finds the contact ON, it allows the circuit continuity through that particular location in the user program logic. Similarly, a Normally Closed input contact tells the PLC to go to the input register and check it for an OFF condition. In other words, the PLC looks at the designated location in the input register to see if it is OFF. If it is OFF, the PLC registers continuity through that contact in the Ladder diagram.

When these contact symbols represent locations in the output register, they provide status reports on the output devices. A Normally Open output commands the PLC to search for an ON condition at a particular output address. The output contact will have continuity if the output device is ON, but will show discontinuity if the device is OFF. A Normally Closed output contact works the opposite way. It has continuity when the output is OFF and discontinuity when the output is ON.

The most common output symbol found on a PLC Ladder diagram is the relay coil. As mentioned above, the relay coil is not really a coil. Instead, it is the symbol used to represent a location in the output register. An energized coil receives a logical 1 (ON) command in its respective output address only when continuity exists through the rung on which it is located. When the rung's continuous logic path is broken, the energized coil will be de-energized. When this occurs, the output associated with the coil will change to a logical 0 (OFF) state.

The de-energized coil acts in the opposite manner. When the power path to a de-energized coil is open, the output associated with the coil will be a logical 1 (ON). When the power path to the de-energized coil is complete, the output will be set to OFF.. Both energized and de-energized coils are also collectively known as output relays.

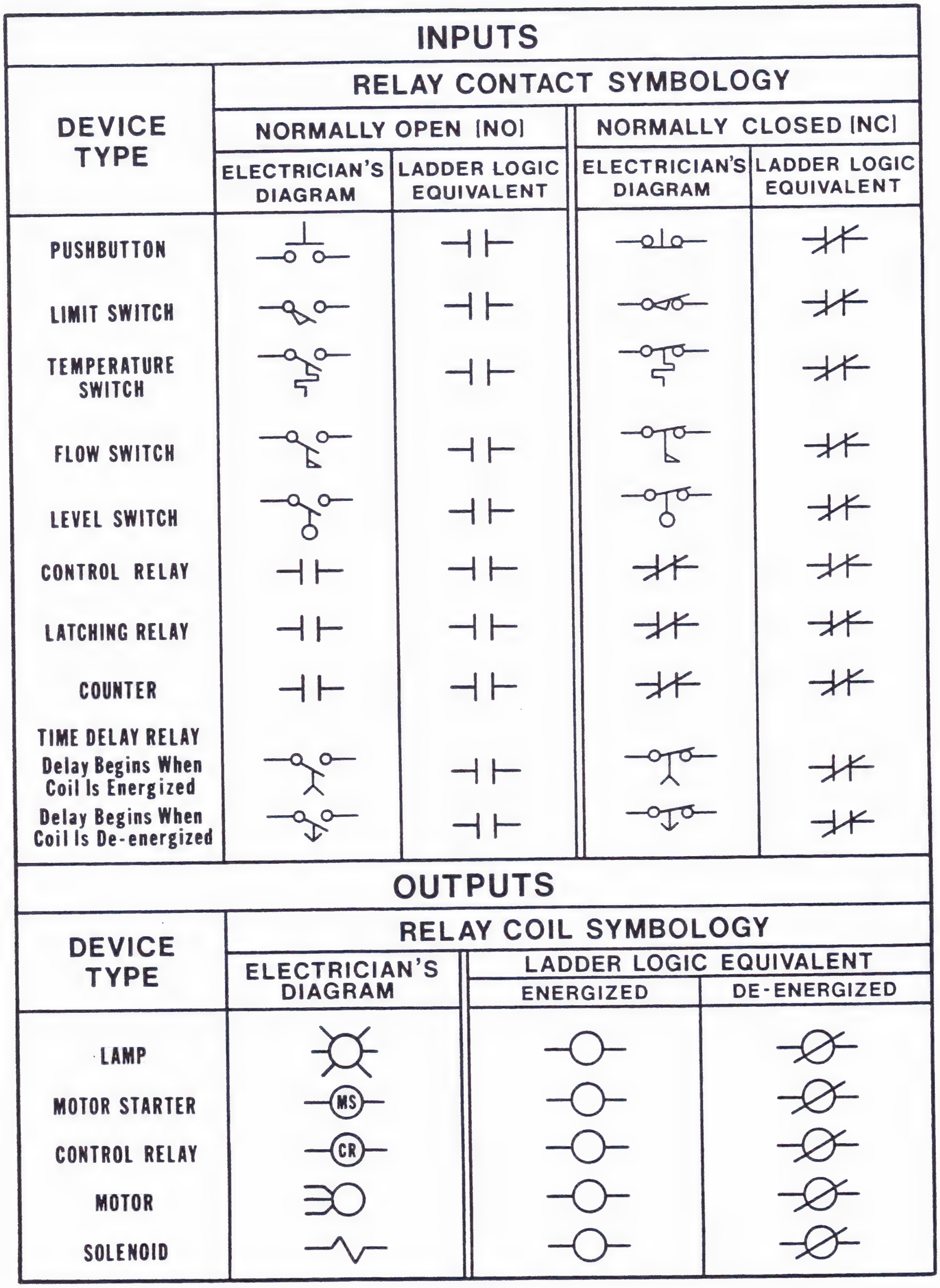


Figure 2.2 Input and Output Symbols

2.4 COMMON CIRCUITRY

In the following pages, we will examine some common circuits and see how they can be transferred into the PLC.

Figure 2.3 shows a circuit with a simple Normally Open momentary pushbutton which, when activated, will energize a relay.

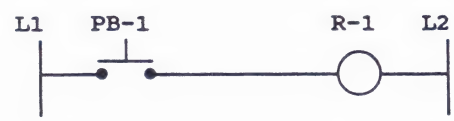


Figure 2.3 Example of a Simple Circuit

The relay (R-1) remains energized for as long as the pushbutton (PB-1) is activated.

In order to get the PLC to perform this action, PB-1 must be wired to the appropriate input, and R-1 wired to the appropriate output. This is shown graphically in Figure 2.4 below.

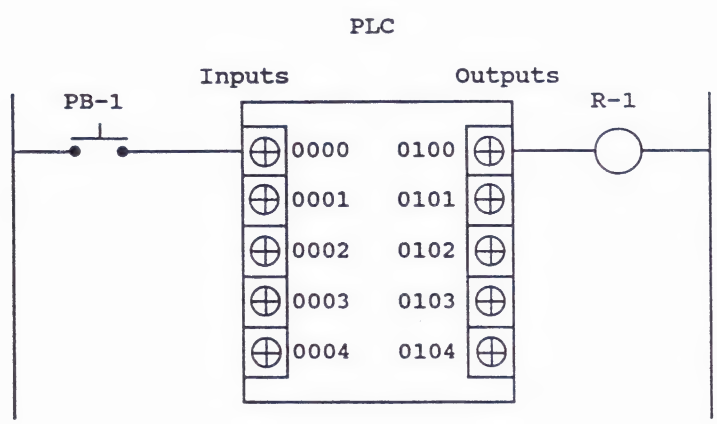


Figure 2.4 Graphic Depiction of a Wired Circuit

Only after each input and output is assigned to the appropriate terminals can we start programming the PLC. Figure 2.5 shows the same circuit in the form of a Ladder Logic diagram in which terminal (I/O) locations are

substituted for the device name (i.e. PB-1, R-1). The PLC, in this case, looks for an ON condition only at Point 00 in order to complete the circuit so Output 0100 (R-1) will energize.

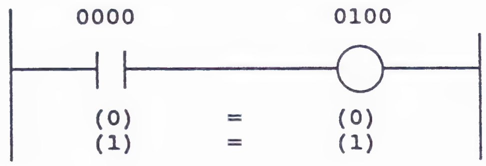


Figure 2.5 Ladder Logic Equivalent

If Input 0000 finds an OFF (0) condition, then Output 0100 is turned OFF. If however, Input 0000 finds an ON (1) condition, then Output 0100 will turn ON.

In the case of Normally Closed logic, the wiring as shown in Figure 2.4 remains the same, (i.e., Normally Open pushbutton which closes upon actuation), but we will invert the logic when we create a new Ladder diagram (see Figure 2.6).

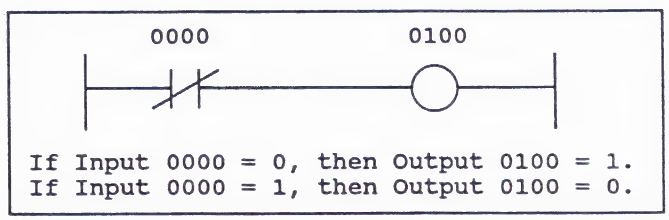
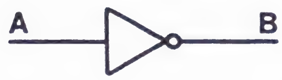


Figure 2.6 Not Logic Circuit

This type of circuit is called a NOT statement. A NOT statement performs an important function in logic circuits. Also called an Inverter, the NOT statement is used to invert the logic level of the signal applied to its input. In other words, the NOT function provides an opposite output. The NOT circuit consists of one input and one output (Figure 2.6) and will always perform the following functions:

(a) If the input equals a logic 1, the output will display a logic 0.

(b) If the input equals a logic 0, the output will display a logic 1.



A	B
0	1
1	0

NOT Gate Symbol **NOT Truth Table**

Figure 2.7 Not Gate Symbol and Truth Table

Referring back to Figure 2.6, note that the NOT logic circuit has a Normally Closed (NC) contact. If Input 0000 finds an OFF (0) condition, then Output 0100 will turn ON (1). If however, Input 0000 finds an ON (1) condition, then Output 0100 will turn Off.

The simple logic circuits outlined so far indicate that logic circuits are the basic building blocks of Programmable Controllers, not to mention computer systems. The following paragraphs introduce Series and Parallel circuits also called AND and OR statements.

2.4.1 Series Circuit (AND Statement)

Each type of logic circuit contributes its own special feature to enable the PLC to recognize signals, transfer signals into data, and use this data to communicate with other vital circuits.

Each individual logic circuit is called a gate and each gate is a binary device containing inputs and outputs. The logic level 1 is related to the binary digit 1 and the logic level 0 is related to the binary digit 0 (1 = ON, 0 = OFF). Thus, 1s and 0s are assigned to the functions of the logic gates and may be transposed into the applicable numbering system for communication. The function of each logic gate is expressed as a relationship between the input and output logic levels.

The first gate we will examine is the AND gate. The AND gate may consist of two or more inputs depending on its design and will always perform the following functions:

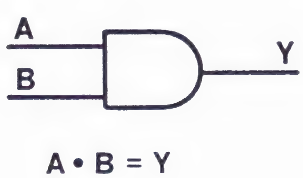
(a) When all inputs equal a logic 1, the output will display a logic 1, thus:

1 and 1 = 1

(b) Any other combination of input logic levels will result in an output of logic 0, thus:

0 and 0 = 0
1 and 0 = 0
0 and 1 = 0

A Truth Table is a diagram showing all possible inputs and the resulting outputs for any given logic circuit (Figure 2.8).



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate Symbol **AND Truth Table**

Figure 2.8 And Gate Symbol and Truth Table

Figure 2.9 illustrates the AND function in an electrical circuit called a Series Circuit.

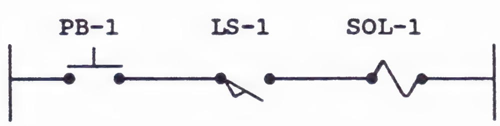


Figure 2.9 Series Circuit

To energize the output (Solenoid-1) both PB-1 <u>AND</u> LS-1 must be actuated.

To solve this logic using a PLC, we must first assign terminals for inputs (to the PLC) and outputs (from the PLC) as shown in Figure 2.10.

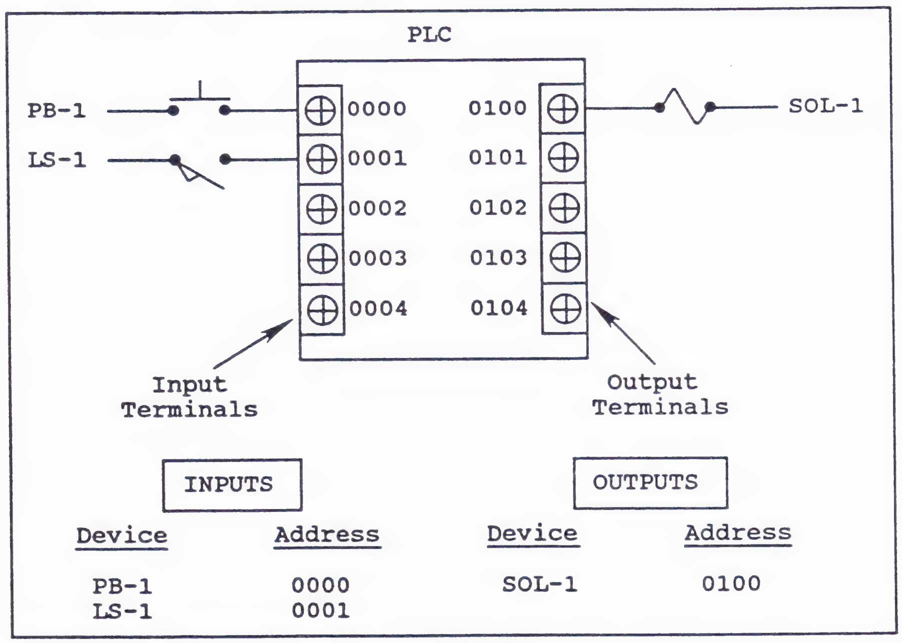


Figure 2.10 Terminal Assignments

Again, we emulate logic by substituting an I/O address for each device name as follows:

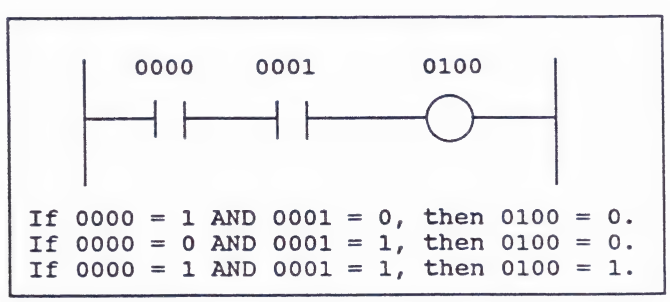


Figure 2.11 Ladder Logic Equivalent

2.4.2 Parallel Circuit (OR Statement)

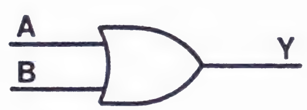
We can use the same techniques described above to solve an OR statement better known in hard-wired terms as a Parallel Circuit.

The OR circuit may also consist of two or more inputs and will always perform the following functions:

(a) If any input equals a logic 1, the output will display a logic 1.

(b) When all inputs equal a logic 0, the output will display a logic 0.

The symbol and Truth Table for the OR gate are shown in Figure 2.12.



$A + B = Y$

OR Gate Symbol

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

OR Truth Table

Figure 2.12 OR Gate Symbol And Truth Table

Figure 2.13 illustrates a Parallel Circuit. In this case, PB-1 OR LS-1 will energize SOL-1.

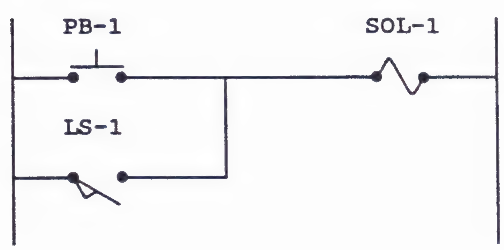


Figure 2.13 Parallel Circuit

Again, we can program this example into the PLC by substituting an I/O address for each device name as shown in Figure 2.14.

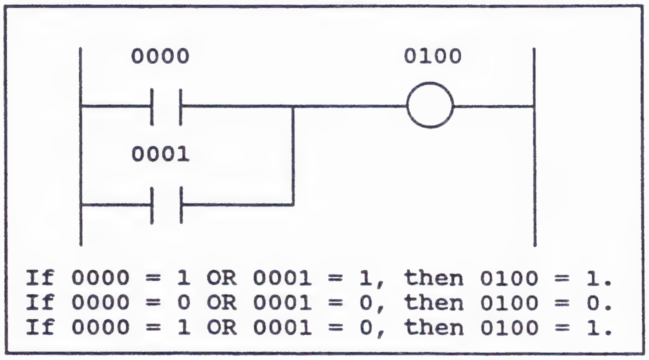


Figure 2.14 Ladder Logic Equivalent

Note that this type of OR statement is also called an "Inclusive OR Statement."

2.4.3 Combined Circuits

AND, OR, and NOT statements can be combined to perform more complex control action. A few examples are provided below.

Example No. 1

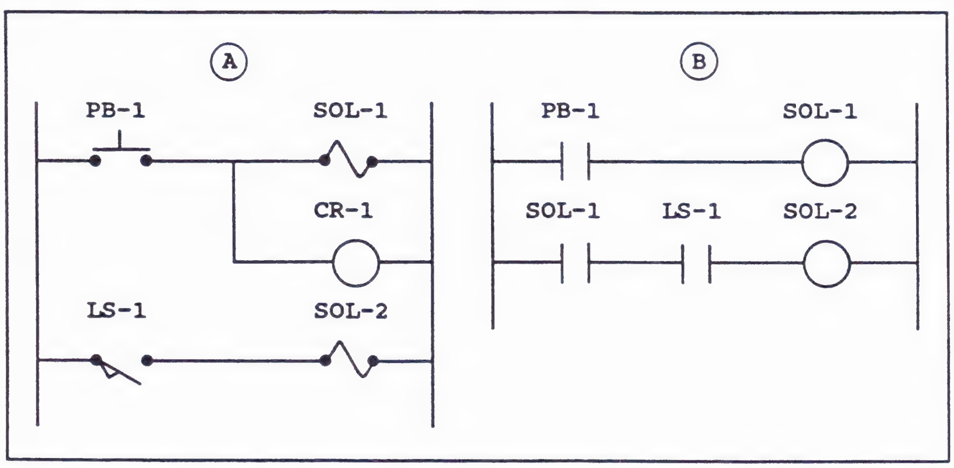


Figure 2.15 Combined Circuits

Note that in part A of Figure 2.15, CR-1 is used as a permissive relay in the second rung. Because we cannot hard-wire SOL-1 into the second rung, we must insert a control relay and use its Normally Open (NO) or Normally Closed (NC) contacts.

Because of the way a PLC operates, we can use the same address that SOL-1 is wired at as an input to any rung, as that address shows the status (OFF or ON; 1 or 0) of SOL-1.

<u>Example No. 2</u>

Figure 2.16 shows a Motor Seal Circuit, also known as a Latch Circuit, E-Stop, Start-Stop, and Keep Circuit.

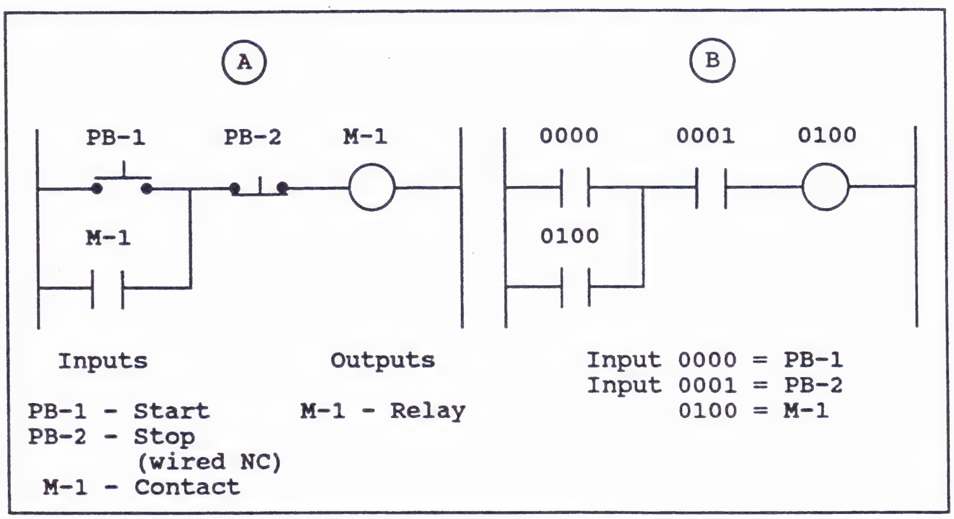


Figure 2.16 Example Of A Motor Seal Circuit

In part A, when PB-1 is activated and PB-2 (Normally Closed) is not activated, Output M-1 will energize. The M-1 contact will also energize creating a latch circuit. Even if PB-1 were to turn OFF, M-1 would still remain energized until PB-2 was activated to open the circuit.

Example No. 3

We previously covered the Inclusive OR statement in which either one input or another (or both) will energize an output. Figure 2.17 depicts what is called an Exclusive OR statement.

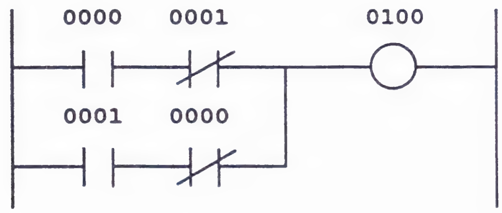


Figure 2.17 Exclusive OR Statement

To energize Output 0100, Input 0000 AND NOT Input 0001 or Input 0001 AND NOT Input 0000 must be activated.

Example No. 4

Omron PLCs may address multiple outputs from a single rung as shown in Figure 2.18.

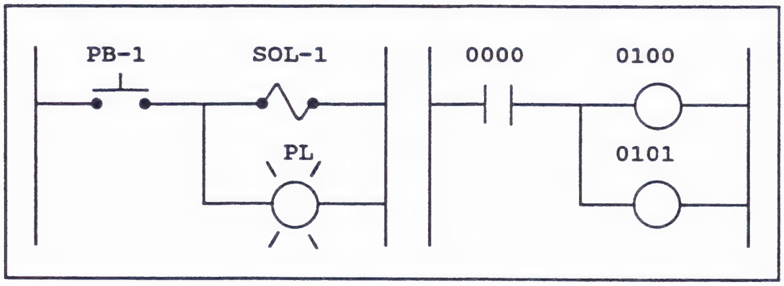


Figure 2.18 Multiple Outputs

In this case, when Input 0000 equals a "1" condition, then Outputs 0100 and 0101 will turn ON (1).

Example No. 5

Some circuits cannot be programmed into the PLC in their present wiring configuration (see Figure 2.19).

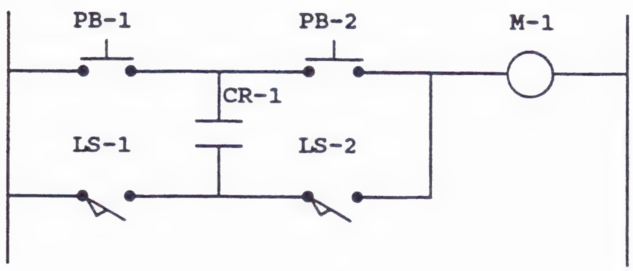


Figure 2.19 Complex Circuit

Since vertical permissives cannot be programmed into the PLC, Internal Auxiliary Relays will have to be used in the Reformatted Ladder diagram. Internal Auxiliary Relays have no ties to the outside world, meaning they cannot control external loads directly and are used only to perform logic inside the PLC. Unused output points can serve as Internal Auxiliary Relays.

The first step in reformatting is to assign I/O addresses for each device as follows:

Inputs	Outputs
PB-1 = 0000	M-1 = 0100
LS-1 = 0001	
PB-2 = 0002	
LS-2 = 0003	
CR-1 = 0004	

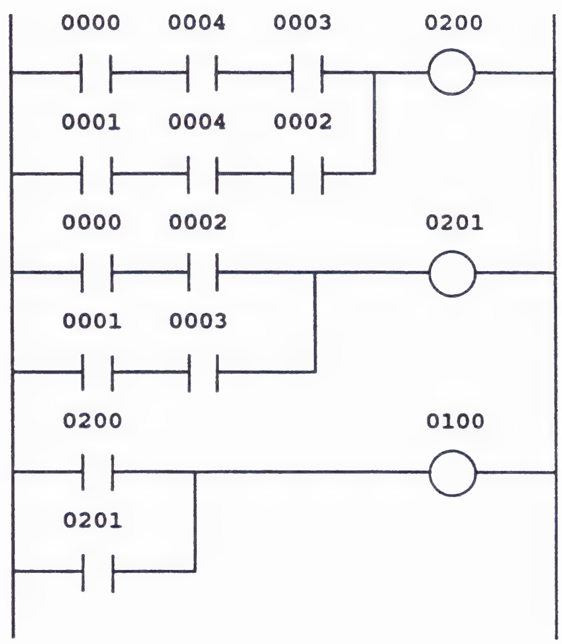


Figure 2.20 Reformatted Circuit

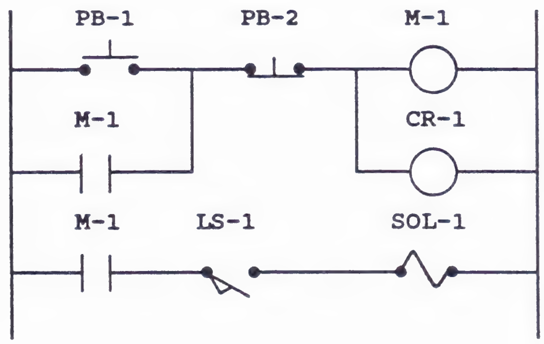
By using unused Outputs 0200 and 0201 as Internal Auxilliary Relays (see Figure 2.20) we can trace which groups of inputs will give us an ON condition at our output. Remember, this action takes place internally in the PLC so no extra wiring is needed.

2.4.4 Practice Exercises

A few exercises are provided below to help you create Ladder Logic diagrams.

Exercise No. 1

Hardwire Diagram



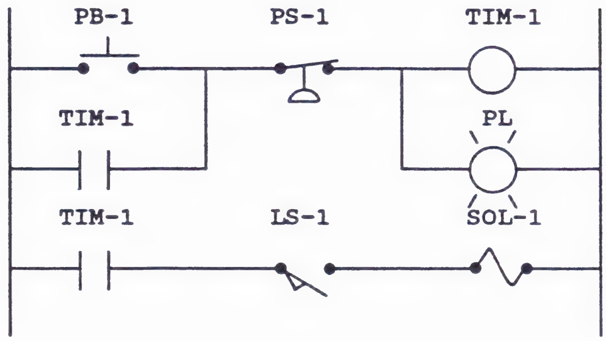
Assigned I/O

Inputs		Outputs	
Device	I/O#	Device	I/O#

Ladder Diagram

Exercise No. 2

Hardwire Diagram



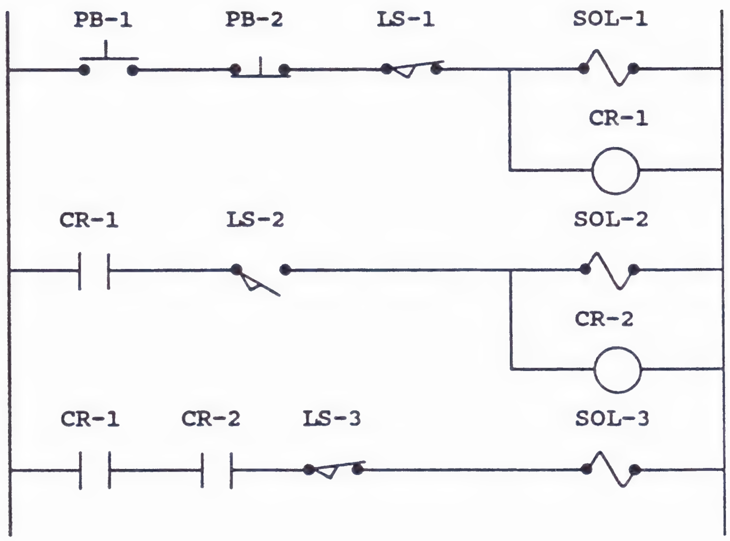
Assigned I/O

Inputs		Outputs	
Device	I/O#	Device	I/O#

Ladder Diagram

Exercise No. 3

Hardwire Diagram



Assigned I/O

Inputs		Outputs	
Device	I/O#	Device	I/O#

Ladder Diagram

- This Page Intentionally Left Blank -

SECTION 3.0
MEMORY MATRIX AND ADDRESSING

3.1 OVERVIEW

This section provides an introduction to memory organization and the principles of Addressing, and is designed to familiarize the user with the Memory Matrix associated with Omron's family of C20K Programmable Controllers.

3.2 MEMORY ORGANIZATION AND ADDRESSING

A Memory Matrix (also referred to as a Memory Map or I/O Map) is a diagram showing a PLC's memory addresses, and programs and data assigned to each section of memory. Sections (areas) are defined by ranges of addresses. For example, there are special memory areas for Timers/Counters, Internal Auxiliary Relays, Data Memory, and so forth. The size of each memory area is measured in terms of words. Each word is composed of a fixed number of bits. For example, the Data Memory area for C20K is: 65 words x 16 bits, while the Timer/Counter area is 48 words x 16 bits. These and other memory areas form special memory blocks that must be addressed by order of their location. A complete Memory Matrix for the C20K is provided in Table 3.1.

An address is a specific location in the PLC's memory. A reference number is assigned to that unique memory location. An address may contain program information (such as instructions) or data. Instructions are generally one (1) to four (4) words long. One instruction is stored in one address regardless of length. Consequently, the maximum number of addresses available changes with the number and kind of instructions used in a program.

Each memory area is defined by a certain address range. The address range refers to a single data area used for I/O points and internal data storage. The address range is accessible in point or channel units. The addresses themselves are therefore expressed in channels or in channel/bit combinations. A channel is comprised of four (4) digits (16 bits) of information. The C20K Family uses a four-digit decimal number to identify an I/O point. The left two digits identify the channel and the right two digits identify the point within the channel.

Figure 3.1 illustrates how the CPU locates addresses. For example, if the address of an input is 0011, the CPU looks for Channel 00 and then scans down to Point 11. Likewise, if the output address is 0104, the CPU locates Channel 01 and then scans down to Point 4.

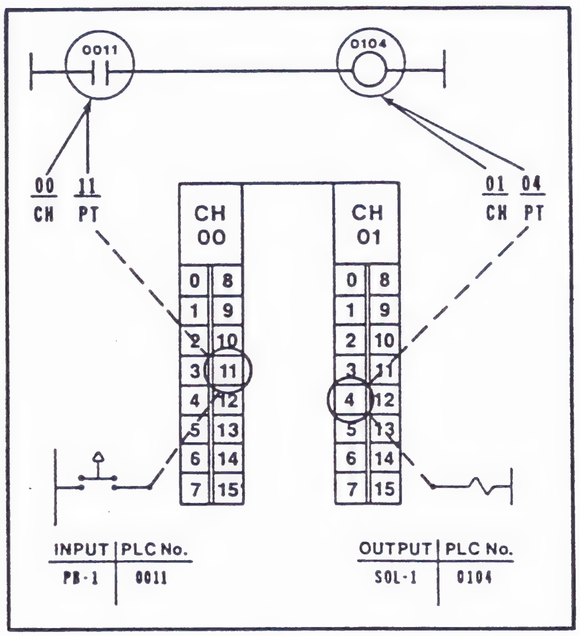


Figure 3.1 Address Location

3.3 MEMORY MATRIX

As stated in the previous paragraphs, there are many memory areas that the user may address. For example, there is the Program (Data) Memory Area where user instructions are stored, executed, and also retained in case of a power failure by using a battery backup unit. All memory areas addressable by the user are known collectively as the Memory Matrix or Memory Map. The following paragraphs describe

these various addressable areas and Table 3.1 provides the applicable addresses for the C20K Family of PLCs.

TABLE 3.1 Memory Matrix For The C20K Family

MEMORY AREA	ADDRESS RANGE	TOTAL ADDRESSABLE CHANNELS	TOTAL ADDRESSABLE POINTS
I/O Table	0000 - 0915	10	148*
Internal Auxiliary Relays	1000 - 1807	9	136
Special Auxiliary Relays	1808 - 1907	NONE	16
Holding Relay Area	HR 0000 - HR 0915	10	160
Temporary Memory Relays	TR0 - TR7	NONE	8
Timers/ Counters	TIM 000 - 47 CNT000 - 47	NONE	NONE
Data Memory Area	DM 00 - DM 63	64	NONE

* Because of addressing architecture only a maximum of 148 I/O is attainable.

CPU	I / O	ADDRESS RANGE
C20K	12 INPUTS	0000 - 0011
	8 OUTPUTS	0100 - 0107
C28K	16 INPUTS	0000 - 0015
	12 OUTPUTS	0100 - 0111
C40K	24 INPUTS	0000 - 0015 0200 - 0207
	16 OUTPUTS	0100 - 0115
C60K	32 INPUTS	0000 - 0015 0100 - 0115
	28 OUTPUTS	0200 - 0215 0300 - 0311

TABLE 3.2 Available I/O Points

3.3.1 Input/Output Table

The Input/Output (I/O) Table is the memory area where real world I/O is addressed. The number of I/O points varies according to which K-type CPU is chosen (see Table 3.1).

NOTES

(1) I/O points 0000 and 0001 serve as the count input and reset input of the High-Speed Counter (HDM) when FUN 61 is used in a program.

(2) The assignment of I/O Channel's starts with Channel 00 (in the CPU). Any unassigned channels from CH02 to 09 can be used as Internal Auxiliary Relay areas.

3.3.2 Internal Auxiliary Relay Area

The Internal Auxiliary Relay Area is the memory area which performs internal logic. This area does not control external devices directly but rather functions as a data process area for performing internal logic with Ladder Logic or numerical values.

NOTE

As with the real I/O Table, the Internal Auxiliary Relay Area does not retain its status after a power failure.

3.3.3 Special Auxiliary Relays

Relays 1808 to 1907 are called Special Auxiliary Relays. They monitor the operation of the PLC and may be programmed as many times as required.

The Special Auxiliary Relays are briefly described as follows:

Relay 1808 - This relay turns ON (is activated) when the supply voltage of the backup battery in the CPU has dropped. The operator may connect the output of this relay to an external indicating

device such as an LED to be alerted of voltage drops.

Relay 1809 - The scan time of the K-type PLCs is 100 milliseconds (ms) or less. If it exceeds 100 ms but is shorter than 130 ms, the ALARM indicator on the CPU lights and Relay 1809 turns ON, but the CPU continues to operate. If the scan time exceeds 130 ms, the ERROR indicator on the CPU lights and the CPU stops.

Relay 1810 - If the High Speed Counter (Fun 61) is used in a program operation, Relay 1810 turns ON for one scan time when the hard reset signal is applied at Input 0001.

Relay 1811 - to 1814 Relays 1811, 1812, and 1814 are normally OFF, whereas Relay 1813 is normally ON. By connecting these relays to external indicating devices such as LEDs, they can be used to monitor the operating status of the PLC.

Relay 1815 - Relay 1815 turns ON for one scan time when the PLC is placed in the RUN mode. It also resets the PLC upon power application.

Relays 1900 - to 1902 These relays are used to generate clock pulses as follows:

Relay 1900 - generates a 0.1-second clock pulse.
Relay 1901 - generates a 0.2-second clock pulse.
Relay 1902 - generates a 1.0-second clock pulse.

They may be used in conjunction with a counter to form an extended timer or a timer which retains data during a power failure. These relays may also be used to form a blinking circuit.

NOTE

The 0.1 second clock pulse produced by Relay 1900 has an ON time of 50 ms. Therefore, if the execution time of the program is too long, the CPU may not be able to read the pulse.

Relay 1903 - Relay 1903 serves as an ERROR flag and turns ON when the result of an arithmetic operation is not output in BCD. It also turns ON if the value of the BIN data processed by the BIN to BCD or BCD to BIN conversion instructions (FUN 23 and 24) exceeds 9,999.

Relay 1904 - Relay 1904 serves as a CARRY flag and operates when a carry is generated as the result of an arithmetic operation such as ADD (FUN 30) or SUBTRACT (FUN 31). Relay 1904 can be forced ON by the Set Carry (FUN 40) instruction or reset by the Clear Carry (FUN 41) flag.

Relays 1905 - to 1907 - These relays function as flags when the Compare (FUN 20) instruction is used in a program. Relay 1905 operates if the Compare result is "more than" (>). Relay 1906 operates if the Compare result is "equal to" (=). Relay 1907 operates if the result is "less than" (<).

3.3.4 Holding Relay Area

The Holding Relay (HR) Area is used for internal data storage and manipulation. This memory area is memory retentive in that it retains data and ON/OFF status if a power failure occurs.

The K-type PLCs contain 160 Holding Relays with 16 relays in each of 10 channels (HR0000-HR0915).

3.3.5 Temporary Memory Relays

Temporary Relays are program instructions used in Ladder diagrams which do not adhere to circuit or syntax rules. They are used in circuits having two or more branched output coils when Interlock (FUN 02) and Interlock Clear (FUN 03) instructions cannot be used.

The C20K family of PLCs contains eight Temporary Memory Relays: TR0 to TR7. Up to eight Temporary Memory Relays can be used in any order. However, the same relay number cannot be duplicated within one block. All eight relays are available again for use in subsequent blocks.

3.3.6 Timer/Counter Area (00-47)

The Timer/Counter Area is a single data area used for Timers, High-Speed Timers, Counters, and Reversible Counters. Because Timers/Counters occupy the same matrix, a counter bearing a certain number cannot be specified as a timer with the same number. For example, CNT 10 and TIM 10 may not be used together.

NOTES

(1) Timer/Counter registers are memory retentive. Set values are retained even if a power failure occurs. However, only the present values of Counters are retained in the event of a power failure.

(2) When the Reversible Drum Counter (RDM:FUN 60) is used, TIM/CNT 46 is used as the present value area and thus cannot be used for any other purpose.

(3) When the High-Speed Counter (HDM: FUN 61) is used, TIM/CNT 47 is used as the present value storage area of the HDM and thus cannot be used for any other purpose.

3.3.7 Data Memory Area

The Data Memory (DM) Area is used for internal data storage and manipulation. This area is accessible in channel units only. Each DM consists of 16 bits.

NOTES

(1) The DM retains data during a power failure.

(2) When an RDM (FUN 60) instruction is used in a program, DM Channels 00 to 31 are used as the upper and lower limit value areas. Therefore, they cannot be used for any other purpose.

(3) When the HDM (FUN 61) instruction is used, DM Channels 32 to 63 are used as the upper and lower limit value areas. Therefore, they cannot be used for any other purpose.

-This Page Intentionally Left Blank -

SECTION 4.0
PROGRAMMING

4.1 OVERVIEW

This section is designed to build on the principles set forth in the previous sections and introduce the user to programming. The trainee will identify program requirements and procede to program basic logic statements before learning how to program Timers, Counters, and other types of function logic. Lastly, the trainee will learn how to store programs onto Cassette Tape and use the P-ROM Writer and Printer Interface units.

4.2 IDENTIFYING PROGRAM REQUIREMENTS

Before programming may begin, several requirements must be considered. These include:

(a) Determining what the control system must accomplish and in what order.

(b) Assignment of Input and Output devices, i.e., determining which external devices will send signals to and receive from the PLC.

(c) Creation of a Ladder diagram.

(d) Coding the Ladder Logic symbols into a form that can be entered into the CPU via the Programming Console.

4.2.1 Assignment Of Input/Output Devices

The most important step in developing any program is determining what the control system must do and in what order the desired task(s) should be accomplished.

Once a task or goal is defined, the components necessary to achieve the results must be identified. Input devices send signals to the PLC, and are designed to sense a particular condition in the environment including temperature, mechanical motion, pressure, and switch position. Typical input devices the user may choose include pushbuttons, limit switches, thumbwheels, selector switches, and sensors.

In contrast, output devices receive signals from the PLC and are designed to change some aspect of their environment, i.e., they get things done. Typical output devices to choose from include motor starters, solenoids, valve actuators, indicator lights, and simple solid-state relays.

4.2.2 Determining Input/Output Relationships

After the user selects the desired input and output devices, Input/Output (I/O) relationships need to be determined. For example, the operator must determine what outputs are to function when switch closures or voltage signals are received from a specific input sensor.

The bulk of Input/Output relationships are comprised of what are called "One to One" relationships. An example of the straight forward One to One relationship is as follows: When Input 0001 is closed, Output 0012 is energized. AND, OR, and NOT logic statements are all functions of One to One I/O relationships.

Counters and timers also play a part in determining I/O relationships. A need for counting arises when several input signals from the same sensor are generated before an output is activated. Counters receive data from an input sensor and receive control signals internally. When the input sensor has signaled the right number of input events, an output is generated. The counters are preset to the number of events desired before an output is activated.

Timers provide internal responses to input closures and provide an output signal bearing a time relationship to the input signal. The desired time is preset and clock pulses are applied to the timer. Counters can be actuated by timers and the effect is the same as multiplying the second counter output by the multiplier for that position. For example, a timer set for 60 seconds can drive a counter set for 60 counts. The counter will be counting minutes while the first timer is counting seconds.

Once I/O relationships have been determined, the operator may begin to assign a particular address to each input and output in the form of a 4-digit number (see Section 3.0).

Output channels that are not being used can function as internal relays. If internal relays are being used, they must be assigned as well. These relays do not control

external devices directly. Instead, they are used as data memory or data process areas to control other relays, timers, and counters. Functionally, these internal auxiliary relays are equivalent to the internal relays used in relay control panels.

At this point, the operator has determined which devices are to be controlled, how they relate to each other, and the sequence of the controlled task.

4.2.3 Creation Of A Ladder Diagram

The next step in program development is the creation of a Ladder diagram. Two vertical lines spaced a few inches apart represent the uprights on the Ladder and signify the power wires or buses. All switching circuits are connected to the left bus, the ungrounded or hot side of the AC power line. All load devices are connected to the right bus, the neutral or grounded side of the AC power line. The rungs (horizontal circuits) are drawn starting with switches connected to the left bus and passing through switch or relay contacts. The circuit continues through a relay coil or load device and terminates at the right bus. The Ladder diagram is usually prepared sequentially in the order the action occurs. The C20K Family of PLCs executes programs according to the sequence in which the instructions were entered and thereby stored in the CPU memory. It is therefore essential to program in the correct sequence.

The following important points serve to aid the user in creating a Ladder Logic diagram:

(1) The number of contacts is not limited for I/Os, Internal Auxiliary Relays, Timers/Ccounters, etc. Use as many contacts as required to configure a simple, clear circuit.

(2) In Ladder diagrams, signals flow from left to right.

(3) Coils cannot be directly connected to the left bus. If necessary, connect the coil through the Normally Closed contact of an unused Internal Auxiliary Relay or use Special Auxiliary Relay 1813 (Normally On) as a dummy.

(4) There is no limit to the number of contacts that can be connected in series or the number of contacts that can be connected in parallel.

(5) All outputs are also provided with auxiliary programmable contacts that can be used in programming. The number of programming contacts that can be used per output is not limited.

(6) No contact can be programmed on the right side of an output coil.

(7) For contact and coil numbers in the circuit, use the I/O numbers described in Table 3.1.

(8) Coil numbers cannot be used in duplicate.

(9) Two or more coils can be connected in parallel.

(10) The program is executed from the first address to the End statement.

4.2.4 Coding Ladder Logic Symbols

After the Ladder diagram is complete, it must be converted into a language the PLC can use. This language consists of addresses, instructions, and data. The addresses are locations in the memory where instructions and data are stored (see Section 3.0). The instructions are used to tell the PLC what to do using the data which follows each instruction.

Once all of the preceding determinations have been made, the operator is ready to begin programming.

4.3 START-UP PROCEDURE (MEMORY AND DATA CLEAR)

After the Programming Console has been connected to the front panel of th PLC, the operator may prepare for programming by performing the following steps:

(a) Set the mode selector on the Programming Console to the PROGRAM position.

(b) Plug in the AC power cord to obtain power. The LCD will display:

```
<PROGRAM>
PASSWORD!
```

(c) Enter the Password by pressing:

CLR, MONTR, CLR

The LCD will display the first address:

```
0000
```

(d) Before entering a new program, the operator must erase previously stored data in the RAM memory. The Holding Relay (HR), Counter (CNT), and Data Memory (DM) areas will also be cleared.

To erase the previous program, press:

(1) CLR
(2) PLAY/SET
(3) NOT
(4) REC/RESET
(5) MONTR

The LCD will show:

```
0000MEMORY CLR
END    HR CNT DM
```

NOTE

If the operator wishes to preserve data in the HR, CNT, or DM areas, the key of the specific area must be pressed before the MONTR key.

4.4 BASIC STATEMENT LOGIC

Once a basic understanding of numbering and logic is acquired, it is important to translate the principles into practical applications. For example, turning on a light switch to illuminate a light bulb illustrates how machine logic relates input actions to output reactions. If a switch is turned ON or "closed", it represents a "1" or "working" condition and the light bulb illuminates. The following paragraphs provide practical applications for the AND, OR, and NOT functions.

4.4.1 AND Statement

Two switches connected in a series provide a good example of an AND statement (Figure 4.1). The logical statement would read: If A and B are ON, the light is ON. Note that switches A and B must both be ON to apply power to the light bulb. This also means that either switch can turn the light OFF.

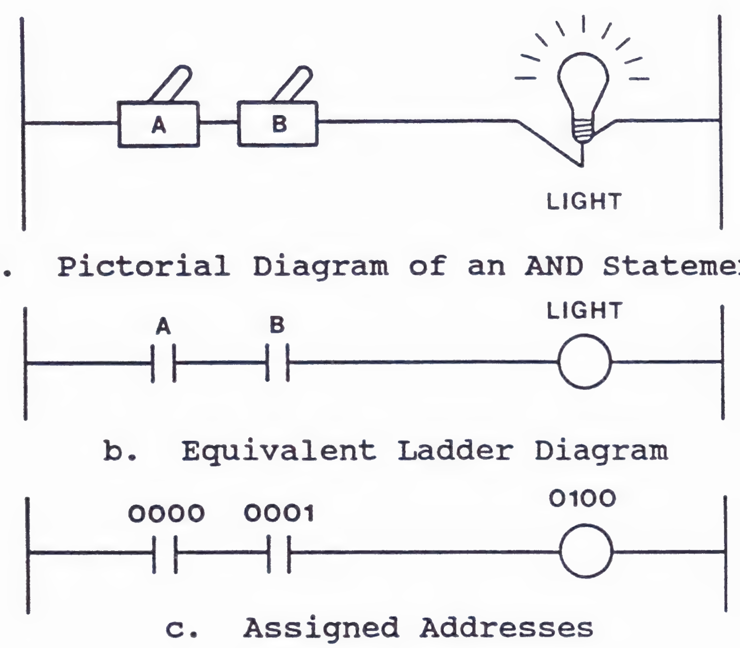


Figure 4.1 Example Of An AND Statement

The operator must assign an address to each input and output (Figure 4.1c) before programming can begin. To review, the first two numbers of an I/O address indicate the channel number while the last two digits indicate the specific point within the channel.

The first item in a rung is entered by pressing the LD key, the first input address, and the WRITE key (note that the CPU does not receive the keyed data until the WRITE key is pressed). The next item to be entered is the logic word AND, the second input address, and the WRITE key. Next, the output is entered by pressing the OUT key, the output address, and the WRITE key. No program is complete without entering an 'End' statement. C-Series controllers use 'FUN, 01, WRITE' to signify the end of a program. To summarize, the step sequence to enter this example of an AND statement is as follows:

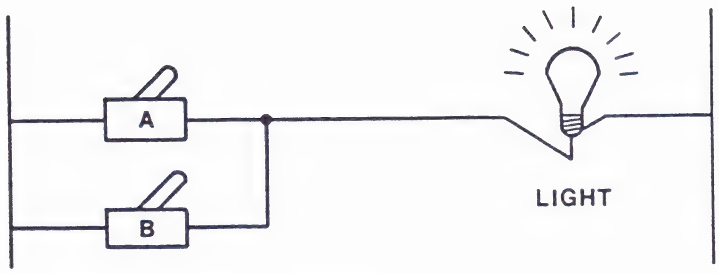
	Address	Key Sequence
(1)	0000	LD, 0000, WRITE (0000 is a default input. On Omron PLCs, no zeros need to be entered.)
(2)	0001	AND, 0001, WRITE
(3)	0002	OUT, 0100, WRITE
(4)	0003	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

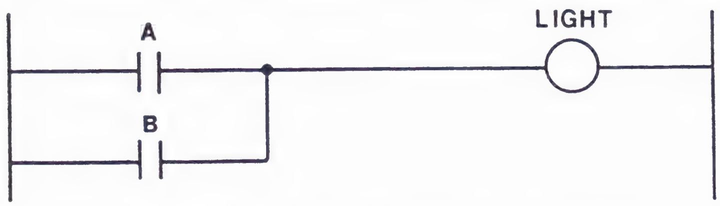
When the switch connected to Input 0000 is closed, AND the switch connected to Input 0001 is closed, the light connected to Output 0100 illuminates. Note that the C20K Family will accept as many AND inputs in a single rung as the operator wishes.

4.4.2 OR Statement

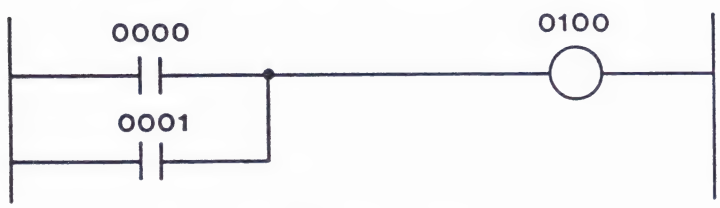
Two switches connected in parallel provide a good example of an OR statement (Figure 4.2). The logical statement would read: If A or B (or both) are ON, the light is ON. Note that Switch A or Switch B can turn the light ON, but both must be turned OFF in order to turn the light OFF.



a. Pictorial Diagram of an OR Statement



b. Equivalent Ladder Diagram



c. Assigned Addresses

Figure 4.2 Example Of An OR Statement

For simplicity, the same address assignments used in Figure 4.1 are used for the inputs and outputs in Figure 4.2.

To program an OR statement, the operator must first erase the previous program by pressing the following keys in sequence.

(1) CLR
(2) PLAY/SET
(3) NOT
(4) REC/RESET
(5) MONTR
(6) CLR

This will be referred to as the "Erase Procedure" from now on and must be performed before entering a new program to ensure that the previous program has been erased.

The display shows 0000 indicating the starting address. The operator may now enter the program for an OR statement as follows:

	Address	Key Sequence
(1)	0000	LD, 0000, WRITE
(2)	0001	OR, 0001, WRITE
(3)	0002	OUT, 0100, WRITE
(4)	0003	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

Remember, when either Switch 0000 or Switch 0001 (or both) are closed, Light 0100 illuminates.

4.4.3 NOT Statement

The NOT statement is used to create Normally Closed (NC) contacts. To review, an NC contact refers to an input or output which is examined for a '0' or OFF condition. In this case, the contact would remain closed to allow current to flow through the contact. If however, the input or output has a '1' condition, the contact would open and not allow current to flow through.

The AND and OR statements in Figures 4.1 and 4.2 respectively, can be modified to use NC contacts as shown in Figure 4.3.

a. Modified AND Statement

b. Modified OR Statement

Figure 4.3 Examples Of A NOT Statement

To program an NC contact, first erase the previous program by performing the Erase Procedure.

To modify the AND statement as shown in Figure 4.3a, enter the following:

	Address	Key Sequence
(1)	0000	LD, 0000, WRITE
(2)	0001	AND, NOT, 0001, WRITE
(3)	0002	OUT, 0100, WRITE
(4)	0003	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

Programming NC contacts for input points reverses the logic. When the external device connected to Input 0001 is closed, the internal contacts are open.

To modify the OR statement as shown in Figure 4.3b, first erase the previous program (p.76). Once the display shows 0000 indicating the starting address, enter the following:

	Address	Key Sequence
(1)	0000	LD, NOT, 0000, WRITE
(2)	0001	OR, NOT, 0001, WRITE
(3)	0002	OUT, 0100, WRITE
(4)	0003	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

Output 0100 remains energized unless both of the external input switches are closed. When closed, the external contacts energize the internal relays 0000 and 0001 to open the circuit for Output 0100.

4.5 AND LD and OR LD STATEMENTS

When a Ladder diagram contains a parallel-series or series-parallel circuit within a rung, an AND LD or OR LD instruction must be introduced. A Ladder diagram containing both an AND and an OR circuit must be separated or divided into segments of strictly series or strictly parallels before it can be entered into the program.

The following paragraphs provide examples of AND LD and OR LD statements.

4.5.1 AND LD

An example of an AND LD statement is depicted in Figure 4.4.

Figure 4.4 AND LD Statement

To program this example, perform the Erase Procedure (p.76) and enter the following:

	Address	Key Sequence
(1)	0000	LD, 0000, WRITE
(2)	0001	OR, 0001, WRITE
(3)	0002	LD, 0002, WRITE
(4)	0003	OR, 0003, WRITE
(5)	0004	AND, LD, WRITE
(6)	0005	OUT, 0100, WRITE
(7)	0006	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

Note that a LD statement is the logical start of a line of logic. The LD statement also signifies the start of a new register. For example, Steps 1 and 2 in the program above are stored in Register 0, and Steps 3 and 4 in Register 1. There can be up to eight registers per rung of logic.

The AND LD instruction in Step 5 serves to connect Registers 0 and 1. In other words, Steps 3 and 4 are ANDed with Steps 1 and 2.

4.5.2 OR LD

An example of an OR LD statement is shown in Figure 4.5.

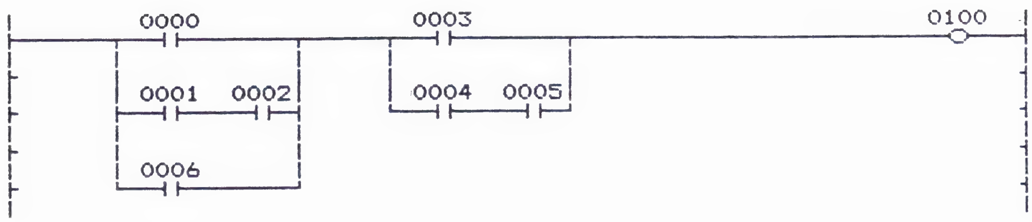


Figure 4.5 OR LD Statement

To program this example, perform the Erase Procedure (p.76) and press the following keys in sequence:

	Address	Key Sequence
(1)	0000	LD, 0000, WRITE
(2)	0001	LD, 0001, WRITE
(3)	0002	AND, 0002, WRITE
(4)	0003	OR, 0006, WRITE
(5)	0004	OR, LD, WRITE
(6)	0005	LD, 0003, WRITE
(7)	0006	LD, 0004, WRITE
(8)	0007	AND, 0005, WRITE
(9)	0008	OR, LD, WRITE
(10)	0009	AND, LD, WRITE
(11)	0010	OUT, 0100, WRITE
(12)	0011	FUN, 01, WRITE

To verify the program, press CLR, CLR, SRCH.

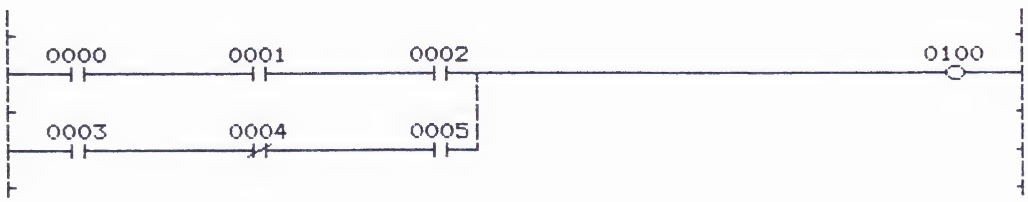
Note the following:

- Register 0 is comprised of Step 1
- Register 1 is comprised of Steps 2, 3, and 4
- Step 5 ORs Register 1 with Register 0
- Register 2 is comprised of Step 6
- Register 3 is comprised of Steps 7 and 8
- Step 9 ORs Register 3 with Register 2

4.5.3 Supplemental Examples of OR LD and AND LD Statements

Example No. 1

Ladder Diagram



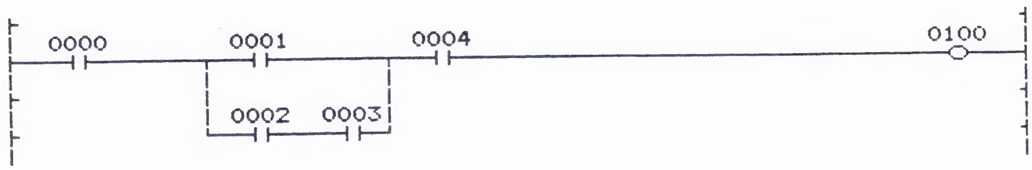
Statement Logic

Address	Key Sequence
0000	LD, 0000, WRITE
0001	AND, 0001, WRITE
0002	AND, 0002, WRITE
0003	LD, 0003, WRITE
0004	AND, NOT, 0004, WRITE
0005	AND, 0005, WRITE
0006	OR, LD, WRITE (ORs Addresses 0000-0002 with Addresses 0003-0005)
0007	OUT, 0100, WRITE
0008	FUN, 01, WRITE

To verify, press: CLR, CLR, SRCH.

Example No. 2

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0000, WRITE
0001	LD, 0001, WRITE
0002	LD, 0002, WRITE
0003	AND, 0003, WRITE
0004	OR, LD, WRITE (ORs Addresses 0002 and 0003 with Addresses 0001)
0005	AND, LD, WRITE (ANDs Addresses 0001-0003 with Address 0000)
0006	AND, 0004, WRITE
0007	OUT, 0100, WRITE
0008	FUN, 01, WRITE

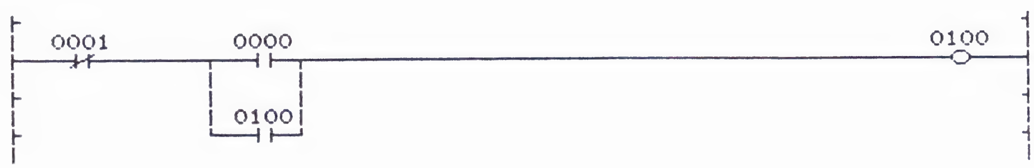
To verify, press: CLR, CLR, SRCH.

Example No. 3

By re-editing a program before actually loading it into the PLC, you may be able to cut down on the amount of AND LD and OR LD statements.

Note the following Ladder diagram and the key sequence to enter the program.

Ladder Diagram

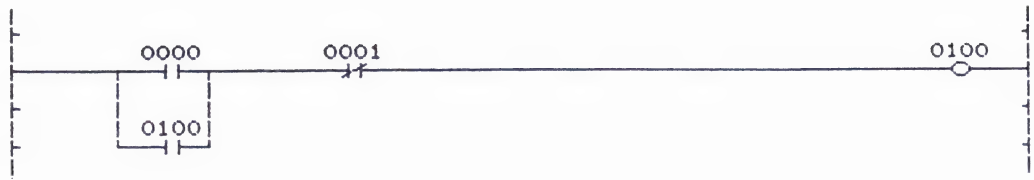


Statement Logic

Address	Key Sequence
0000	LD, NOT, 0001, WRITE
0001	LD, 0000, WRITE
0002	OR, 0100, WRITE
0003	AND, LD, WRITE
0004	OUT, 0100, WRITE
0005	FUN, 01, WRITE

Now note the reformatted Ladder diagram:

Reformatted Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0000, WRITE
0001	OR, 0100, WRITE
0002	AND, NOT, 0001, WRITE
0003	OUT, 0100, WRITE
0004	FUN, 01, WRITE

The same inputs will turn ON Output 0100 and we save one step (address).

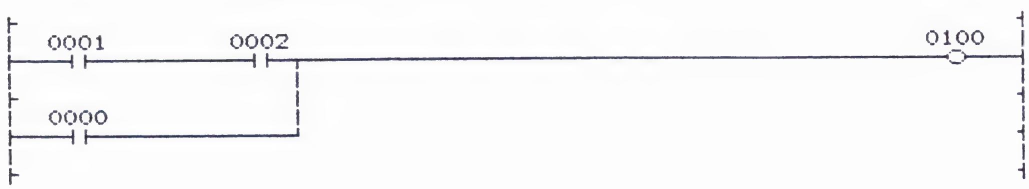
Example No. 4

Ladder Diagram

Statement Logic

Address	Key Sequence
0000	LD, 0000, WRITE
0001	LD, 0001, WRITE
0002	AND, 0002, WRITE
0003	OR, LD, WRITE
0004	OUT, 0100, WRITE
0005	FUN, 01, WRITE

Reformatted Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	AND, 0002, WRITE
0002	OR, 0000, WRITE
0003	OUT, 0100, WRITE
0004	FUN, 01, WRITE

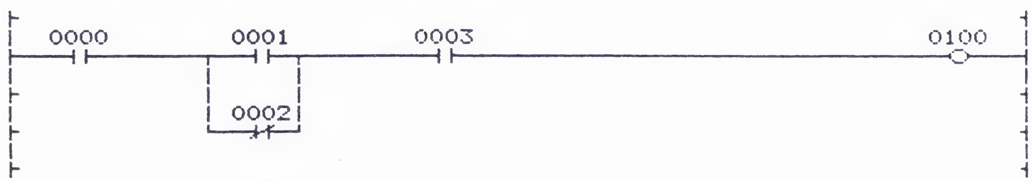
The program does not change the way it turns on the output and again we save one address (step).

4.5.4 Practice Exercises

The following exercises are designed to help develop skills in writing statement logic.. The Ladder diagrams provided may be reformatted as long as the logic does not change. Fill in the 'Address' and 'Key Sequence' portions under "Statement Logic". After writing the program, load it into the PLC to verify.

Exercise No. 1

Ladder Diagram

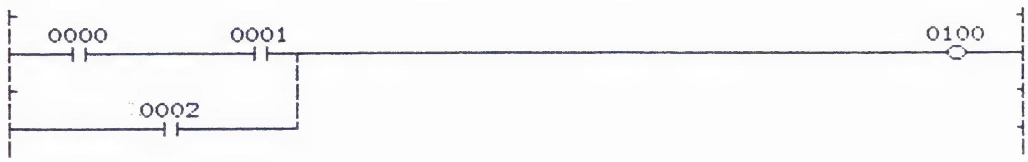


Statement Logic

Address | Key Sequence

Exercise No. 2

Ladder Diagram

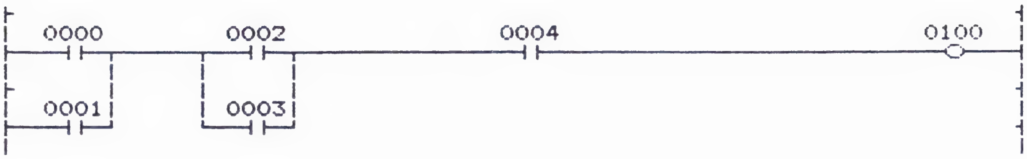


Statement Logic

Address **Key Sequence**

Exercise No. 3

Ladder Diagram

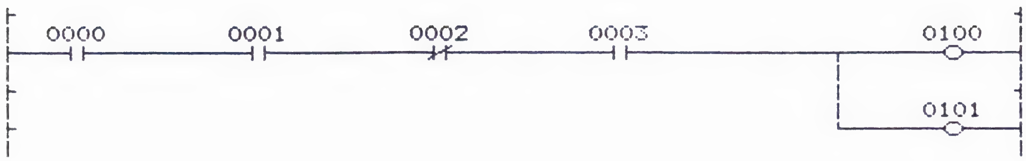


Statement Logic

Address Key Sequence

Exercise No. 4

Ladder Diagram



Statement Logic

Address | Key Sequence

4.6 INSERTIONS AND DELETIONS

There may be instances when the operator accidentally omits an instruction during programming. For example, the operator may have forgotten to insert a set of Normally Closed contacts for Input 0006 between the AND, NOT, 0005 instruction and the contacts for the OUT, 0100 instruction (Figure 4.6).

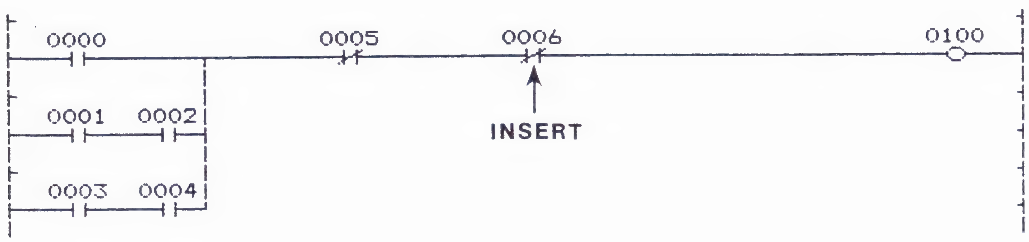


Figure 4.6 Example of an Insert Instruction

To insert the Normally Closed contacts, first set the mode selector switch to the PROGRAM mode. Then press:

OUT, 0100, SRCH

The Programming Console scans through the program until it locates the OUT, 0100 instruction. The address will appear in the upper left hand corner of the display. To insert an instruction ahead of the OUT, 0100 instruction, the operator must enter the following sequence of keys:

AND, NOT, 0006, INS, Down Arrow

The program will automatically move the rest of the instructions down one address to make room for the inserted instruction.

Likewise, there may be instances when the operator wishes to delete an instruction. For example, the operator may wish to delete the AND, NOT, 0006 statement. At this point, the operator would press:

AND, NOT, 0006, SRCH

The console scans through the program until it locates the instruction. The address will appear in the upper left hand

corner of the display. To delete this instruction, the operator must enter the following sequence of keys:

DEL, Up Arrow

The program will automatically move the rest of the instructions up one address.

4.7 ON/OFF-LINE OPERATIONS

In the previous sections, the various procedures for system set-up, PLC start-up procedure, and entering programs have been presented. This section will focus on the various ON/OFF-line operations such as searching and monitoring which the operator can perform using the Programming Console.

Before performing ON/OFF-line operations, make sure the Programming Console is connected to the PLC. The current mode status is displayed in brackets in the upper left hand corner of the LCD display.

```
<PROGRAM>
PASSWORD!
```

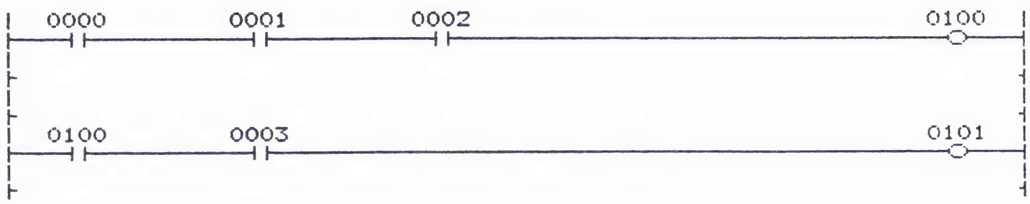
At this point, the PLC will not allow a mode change (RUN, MONITOR, PROGRAM) or allow a program to be viewed until the key sequence CLR, MONTR, CLR is entered. Note that the last mode the Programming Console was in prior to this operation will be the mode displayed regardless of the current switch position.

4.7.1 Program Search

This operation will show the operator how to find (search) a particular program entry in the program. A program search may be accomplished in any of the three modes (RUN, MONITOR, or PROGRAM).

First enter the sample program shown in Figure 4.7.

<u>Ladder Diagram</u>



<u>Statement Logic</u>

Address	Key Sequence
0000	LD, 0000, WRITE
0001	AND, 0001, WRITE
0002	AND, 0002, WRITE
0003	OUT, 0100, WRITE
0004	LD, 0100, WRITE
0005	AND, 0003, WRITE
0006	OUT, 0101, WRITE
0007	FUN, 01, WRITE

To verify, press: CLR, CLR, SRCH.

Figure 4.7 Program Illustrating Search Procedure

To ensure that the program is loaded correctly into the PLC, perform the program check function by pressing:

CLR, CLR, SRCH

This procedure checks to make sure that the PLC <u>understands</u> and can <u>perform</u> the functions written to it. The PLC checks for various errors including circuit error, coil duplication, missing END statement, etc. (See Section 4.8 for an in-depth discussion on errors).

NOTE

> The CLR key serves two purposes:
>
> (a) Pressing CLR once clears the display.
> (b) Pressing CLR twice clears the display and defaults to user address 0000.

You can also step through the program to check entries by pressing the CLR key twice and then the Down Arrow key for each address you wish to check. For example, if you want to check the contents of Address 0004, you would press:

CLR, CLR, and Down Arrow four times

The Down Arrow key increments you through each program address, one address at a time. The Up Arrow key decrements you through the program to the previous address.

NOTE

> Because 0000 is the starting address, if the Up Arrow key is pressed at Address 0000, a double beep will sound indicating a key-in error.

To gain access to a specific step in a program, the operator may also preset an address before actually stepping through the program. For example, press:

CLR, CLR, 0002, Down Arrow

The display will show:

```
0002READ
AND           0002
```

This allows the operator to begin the search from Address 0002. This procedure is called Address Setting.

NOTE

```
0002READ OFF
AND           0002
```

> In the RUN or MONITOR mode, the upper right hand corner of the display will show the program status (OFF, ON) of the I/O in that particular program address.

4.7.1.1 Instruction Search

To gain access to an instruction, specify the instruction and press the SRCH key. For example, to find 'AND, 0003' press:

AND, 0003, SRCH

The PLC will search for the program address of the AND instruction and the display will show:

Program address where 'AND, 0003' is located →

```
0005SRCH
AND          0003
```

NOTES

(1)	If 'AND, 0003' is not found in the program, the search will cease at the address where the End statement is located.
(2)	If 'AND, 0003' is used more than once in a program, the operator may press the SRCH key repeatedly to display successive addresses which contain the instruction.

4.7.1.2 Contact Search

This operation searches for the contact numbers used in a program. For example, to search for Contact 0002 (using the same program keyed in in Figure 4.6) press:

SHIFT, CONT/#, 0002, SRCH

The Programming Console will display the address where Contact 0002 is located (Notes 1 and 2 above also apply for this operation.)

Example No. 2

To search for Contact 0100, press:

SHIFT, CONT/#, 0100, SRCH

The display will show the address where 0100 is used as a contact (not as a coil!)

4.7.1.3 Coil Search

This operation searches for output coils. Using the same program (Figure 4.6) press the following keys to search for Coil 0100:

$\underbrace{\text{CLR, CLR,}}_{\text{Defaults to Address 0000}}$ OUT, 0100, SRCH

The display will show:

```
0003SRCH
OUT            0100
```

See Notes 1 and 2 in Paragraph 4.7.1.1 as they also apply in this operation.

General Note

> While performing any of the preceeding searches (Instruction, Contact, and Coil Search) it is advisable to start the search at Address 0000 so that the PLC will not overlook any entries.

4.7.2 On-Line Monitoring

This operation allows the user to monitor any internal or external I/O point, timer, counter, etc., as well as its status (ON or OFF).

To monitor 'On-Line', perform the following:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press: CLR, CLR. This defaults to Address 0000 and clears the display.

(3) Make sure all switches are turned OFF on the I/O simulator (no LED indicators on the I/O terminal).

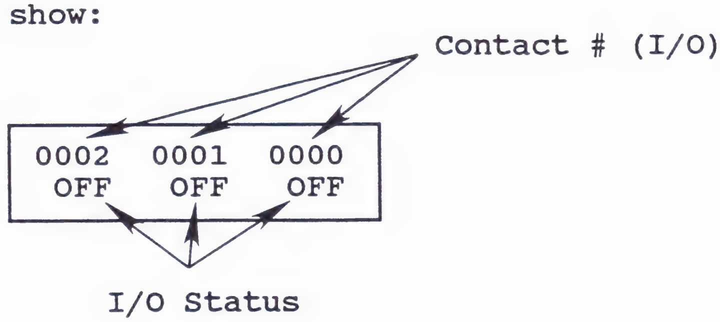
(4) Key in the following:

(a) SHIFT, CONT/#, 0000, MONTR

(b) SHIFT, CONT/#, 0001, MONTR

(c) SHIFT, CONT/#, 0002, MONTR

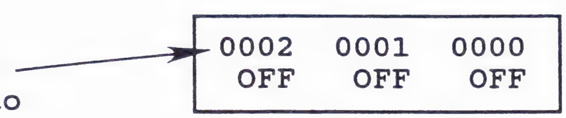
The display will show:



At this point, you can also monitor 0003, 0004, etc. by pressing the Down Arrow or Up Arrow keys.

Only the contact in the upper left hand corner will increment/decrement to the next/previous I/O number.

By pressing the Down Arrow, 0002 will increment to 0003. Pressing the Up Arrow will decrement to 0002.



Now turn on Input 0000 by activating the I/O simulator. You will see the status of Input 0000 change to ON on the display.

NOTES

(1) After starting this multi-point monitor operation (by keying SHIFT, CONT/#, I/O number, MONTR) you only need to enter the numerical values of the subsequent I/O numbers followed by the MONTR key.

(2) A total of six I/O may be monitored at the same time. Three are displayed on the LCD. The other three are contained in the buffer and may be displayed by pressing the MONTR key.

4.8 ERROR AND ERROR CODES

The C20K Programmable Controller has self-diagnostic capabilities to identify many errors. Errors are displayed on the CPU's LEDs and the Programming Console attached to the CPU (see Figure 4.8).

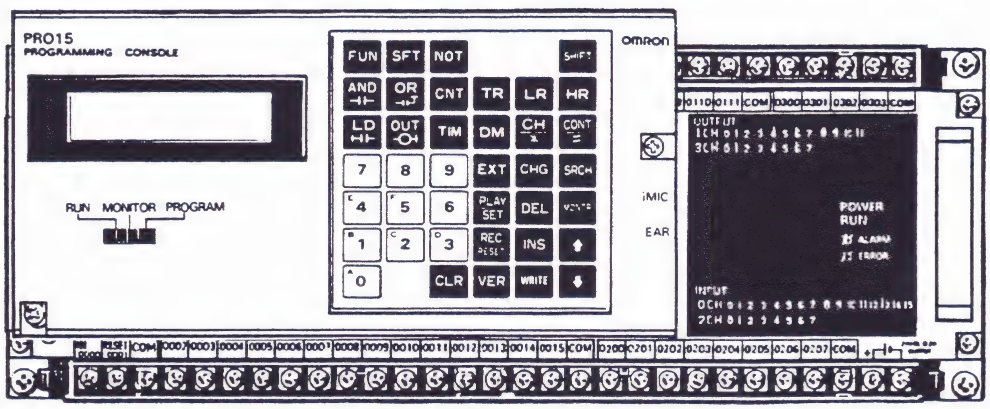


Figure 4.8 Programming Console and CPU

When an error occurs during program execution, an Error or Alarm LED will illuminate on the CPU. The nature of the error can be displayed on the Programming Console by pressing: CLR, FUN, MONTR. To clear this error message, press the MONTR key again. Should more than one error occur simultaneously, the next error message will be displayed after the MONTR key is pressed. Up to three errors/alarms can be viewed. Continue pressing the MONTR key until all of the error messages have been cleared and the display shows:

```
0000 ERR CHK
OK
```

Figure 4.9 shows errors and alarms, and their effect on CPU operation.

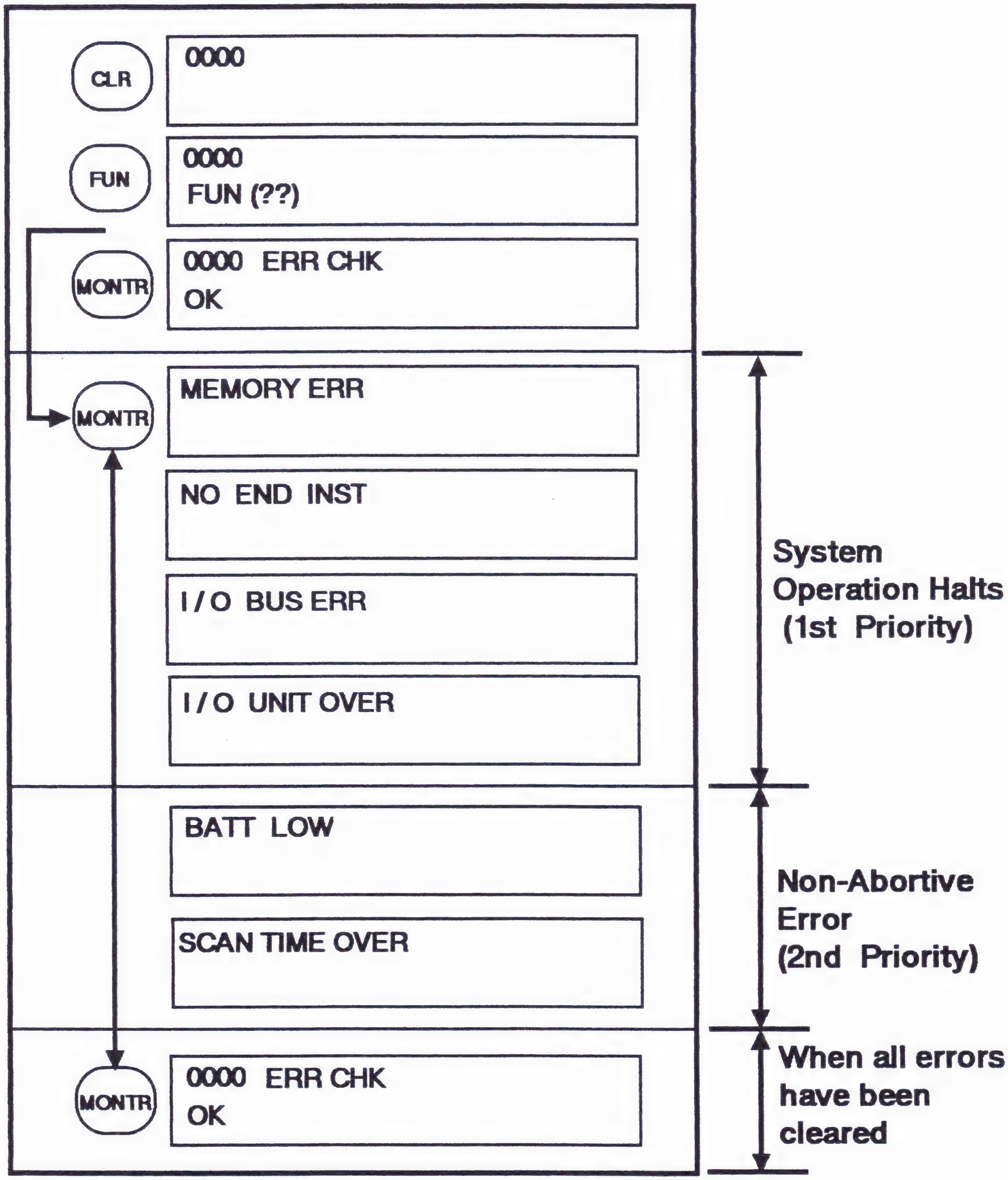


Figure 4.9 Error and Alarm Displays

The following paragraphs will describe three types of errors:

(a) System Errors
(b) Errors During Programming
(c) Program Errors

4.8.1 System Errors

(a) Power Failure - Check the power supply, wiring, etc. The unit will ignore a power failure for 10 ms.

(b) CPU Failure - Hardware error.

(c) Memory Err - An error exists in the PLC memory (RAM). Either there is a loss of memory (partial or full) due to conditions in the CPU, or the settings on the user dipswitch are incorrect.

(d) I/O Buss Err - Communications failed between the CPU and local or remote I/O.

(e) Batt Low - Back-up battery is low and should be replaced.

(f) Scan Time Over - Watchdog timer has timed out and more than 100 ms have elapsed in the program scan.

(g) No End Instr - No FUN 01 instruction is found in the user program before Address 1193 (the last program address).

4.8.2 Errors During Programming

(a) REPL ROM - The CPU dipswitch is set to ROM memory but either no chip or an incorrect one is mounted.

(b) ADDR OVER - The program address exceeds the end address (1193) of the program memory.

(c) <u>SET DATA ERR</u> - A constant exceeding the pre-determined range of an instruction has been used. For example, a timer value has been incorrectly set to #FFFF.

(d) <u>I/O NO ERR</u> - An attempt has been made to enter I/O data (address) which exceeds the pre-determined range. For example, an I/O channel address of 0117, has been entered in a program when Channel 0115 is the maximum range.

(e) <u>PROG OVER</u> - The program exceeds memory capacity. (maximum address range is 0000 - 1193)

4.8.3 Program Errors

Program errors may appear during a program check (CLR, CLR, SRCH in the PROGRAM mode), or when monitoring a program.

(a) <u>????</u> - The program address has been destroyed due to conditions such as high electrical noise, loss of battery back-up, etc. The program (or steps of the program) must be reloaded.

(b) <u>CIRCUIT ERR</u> - There is a logical error in the circuit configuration. The PLC will display the last output where the error exists. For example:

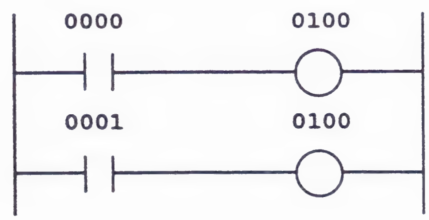
```
LD, 0000, WRITE
LD, 0001, WRITE
OUT, 0100, WRITE
```

When a program check is performed the display will show:

```
0002CIRCUIT ERR
OUT        0100
```

(c) <u>COIL DUPL</u> - The same coil number is assigned to more than one OUT instruction.

Coil duplication presents a problem to the PLC because of the way it scans the user program register (user addresses). For example,



Only the logic in the second rung will be performed. The first rung will not be solved.

NOTE

When errors are displayed, the highest priority errors which cause the CPU to halt are displayed first, regardless of when they occurred (refer back to Figure 4.8).

4.9 FUNCTION LOGIC

The C20K Family of PLCs provide the user with a large selection of programming instructions. These may be categorized into four types including Basic, Data Manipulation, Arithmetic, and Application instructions.

To review, the Basic programming instructions include LD, OUT, AND, OR, NOT, and END. These instructions encompass the basic program control structure of the PLC and are indispensable in almost any program. See Table 4.1 below and refer back to Section 4.4 for programming examples for these basic instructions.

Table 4.1 Basic Instruction Operation

LD	Starts each logic line or block.
OUT	Indicates an output bit.
AND	Performs a logical AND operation on two inputs.
OR	Performs a logical OR operation on two inputs.
NOT	Inverts whatever is before it; often used to form a Normally Closed (NC) input. NOT can be used with LD, OUT, AND, or OR. NOT is also used when programming differentiated instructions.
END (01)	Indicates the end of the program.

The remaining instructions comprise what is termed "Function Logic" and are entered on the Programming Console by pressing the FUN (Function) key and the appropriate numerical value. Timer, Counter, Temporary Relay, etc., instructions are entered by pressing the corresponding key on the Programming Console. Note that the WRITE key must always be pressed to complete the entry of any instruction.

To enter a hexadecimal number, use the numeric keys to enter numerals 0 to 9. To enter A to F (11-15), hold down the SHIFT key and press the appropriate numeric key using the letters superscripted on the upper left of the numeric keys as a guide.

To enter a constant, the CONT/# key must be pressed before specifying the constant.

The following pages contain explanations and programming examples for several functions as well as Timers and Counters. The remainder of the functions are covered in the C20K Advanced course.

<u>FUN 00 (NOP)</u>

This function indicates No Operation (NOP). When the PLC sees this in a user register, it will perform no logical operation to that register. That register lacks any control defined instruction.

When a user program is cleared, the PLC automatically writes a NOP to all registers that have been erased. (If the register contains something other than a user operation or a NOP, refer to Section 4.8 on Error and Error Codes.)

Because NOP is called out as a function (FUN 00), it can also be programmed by the user. For example, anticipating system expansion, the user may wish to save a program register for use at a later date.

NOP may also be used for troubleshooting to allow the operator to "mask" a particular function.

Program Example		Same Program with NOP
Address	Key Sequence	Key Sequence
0000	LD, 0000, WRITE	LD, 0000, WRITE
0001	AND, 0001, WRITE	FUN, 00, WRITE
0002	AND, 0002, WRITE	AND, 0002, WRITE
0003	OUT, 0100, WRITE	OUT, 0100, WRITE
0004	FUN, 01, WRITE	FUN, 01, WRITE

[If Inputs 0000 and 0001 and 0002 = "1" (ON) then Output 0100 = "1" (ON)]

[By using NOP, the user has eliminated the 'AND 0001' statement without losing program space]

For permanent address removal, see Section 4.6 on Insertions and Deletions.

FUN 01 (End Statement)

This function indicates an "End" statement. An End statement must reside in all programs as the PLC will not execute a program lacking an End statement.

NOTE

Even if there is a program written after the End statement, it will not be executed because the PLC stops scanning registers at a FUN 01 statement.

The use of multiple FUN 01 statements may be useful in start-up and system debug operations. A good example of this is shown in Figure 4.10 where the three blocks (A,B,C) represent three machine functions: Start-up (initialization), operation, and finished product.

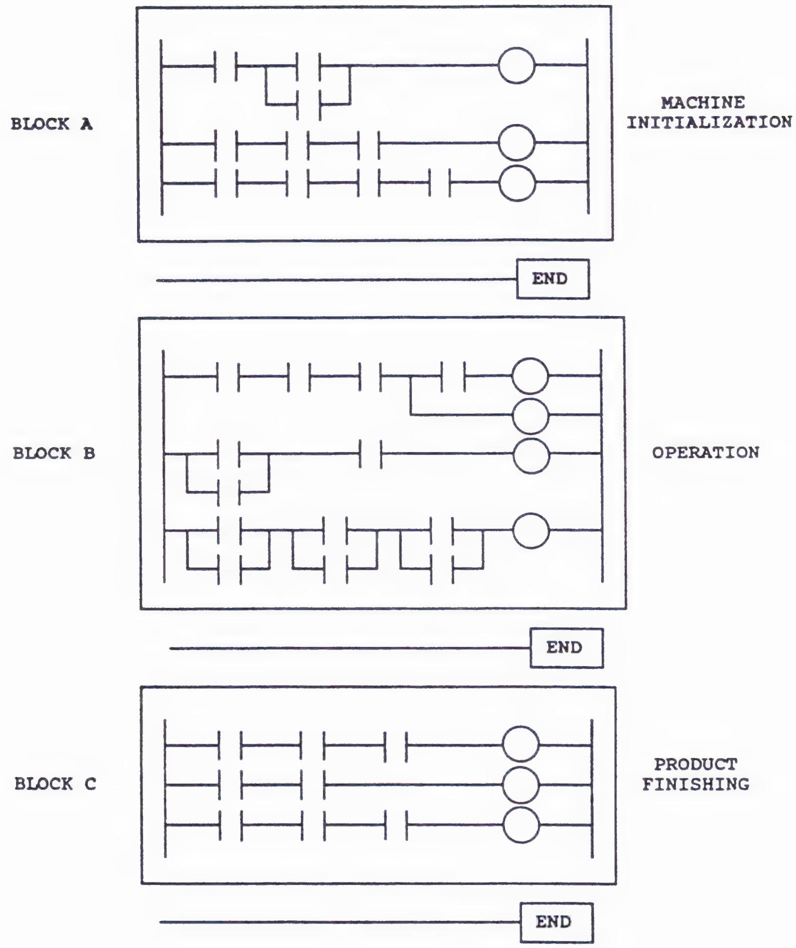


Figure 4.10 Using Multiple END Statements

By inserting an End statement after Block A, the operator can observe, run, or troubleshoot only the Machine Initialization portion of the program. After completion, the operator can remove the End statement (Delete or NOP) and run Blocks A and B together. After troubleshooting and debugging Blocks A and B together, the operator may remove the End statement after Block B and run the complete machine operation (Blocks A, B, and C). Thus, by using multiple End statements, system start-up and debugging are easier to perform since the operator is focusing on only one machine operation at a time.

FUN 02 AND FUN 03 (Interlock and Interlock Clear)

These functions signify an Interlock (IL) and Interlock Clear (ILC) instruction respectively. By using the IL and ILC functions, the operator can mimic the original hardwire style of machine control without needing to use an external electromechanical interlock (see Figure 4.11 below).

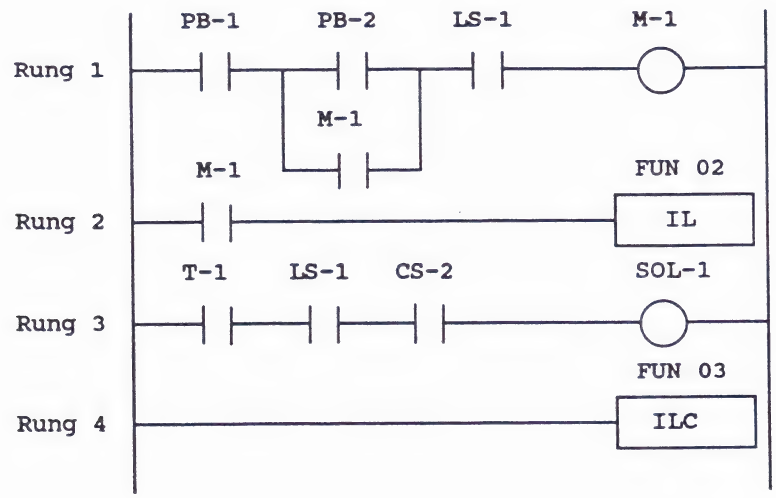


Figure 4.11 IL And ILC Statements

IL and ILC instructions must always be used in pairs. They comprise a control boundary for what is controlled in the IL statement.

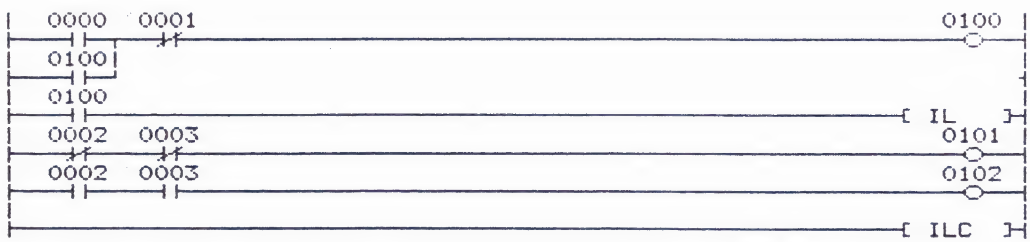
In Figure 4.11, note that the M-1 contact is the relay "permissive". If this contact equals a "1" (ON) condition, the logic in Rung 3 will be performed. If however, the contact equals a "0" (OFF) condition, the logic will not be performed. In other words, the IL instruction will turn ON all of the output coils between the IL and ILC instructions if the conditions indicated before IL are met, and will turn OFF the output coils if the conditions are not met.

NOTES

(1)	In the case of outputs, timers, etc., if the Interlock permissive is "0" (OFF), the outputs are turned OFF and timers are reset. Counters, shift registers, and latching relays are however unchanged.
(2)	There is no limit to the amount of Interlock statements used in a single program.

Example No. 2

The short program in Figure 4.12 uses the Interlock statement to stop the logic being performed if a stop routine is initiated.



Statement Logic

Address	Key Sequence
0000	LD, 0000, WRITE
0001	OR, 0100, WRITE
0002	AND, NOT, 0001, WRITE
0003	OUT, 0100, WRITE
0004	LD, 0100, WRITE
0005	FUN, 02, WRITE
0006	LD, NOT, 0002, WRITE
0007	AND, NOT, 0003, WRITE
0008	OUT, 0101, WRITE
0009	LD, 0002, WRITE
0010	AND, 0003, WRITE
0011	OUT, 0102, WRITE
0012	FUN, 03, WRITE
0013	FUN, 01, WRITE

Figure 4.12 Example No. 2 Of IL/ILC Statements

As mentioned earlier, IL and ILC statements can simulate the same control functions of an electrical or mechanical interlock function in a control process. For example, Figure 4.13 shows an Electrical Ladder diagram in which an electrical interlock is used in a machine process to prevent the operation of any further group of logic.

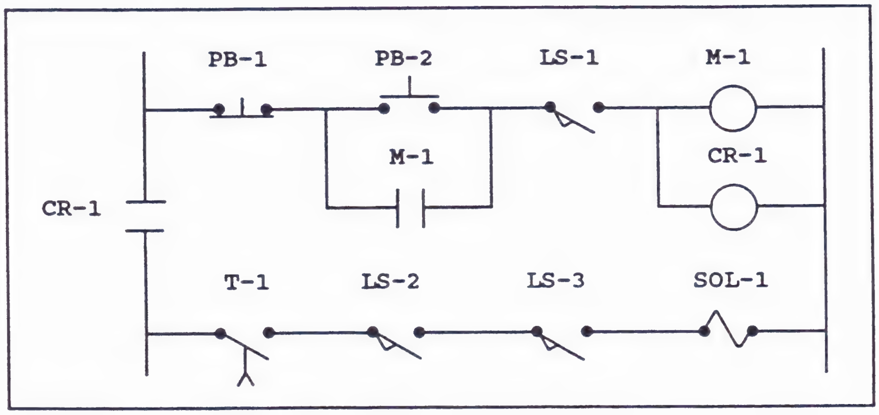
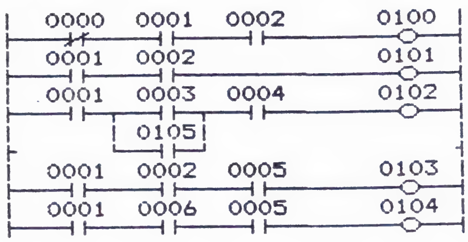


Figure 4.13 Electrical Ladder Diagram Depicting An Electrical Interlock

Note that unless CR-1 is energized, the rest of the machine operation will be locked out.

Interlock statements can also be used as programming aids to save the amount of program addresses used. For example, see Figure 4.14 which contains a program before and after inserting an IL statement. The first program shows Input 0001 used several times. By reformatting the program to include IL and ILC statements, three program addresses are saved, and the program becomes easier to troubleshoot and debug.

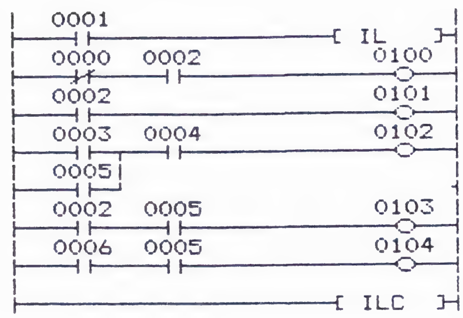
Before IL Is Used



Statement Logic

Address	Key Sequence
0000	LD, NOT, 0000, WRITE
0001	AND, 0001, WRITE
0002	AND, 0002, WRITE
0003	OUT, 0100, WRITE
0004	LD, 0001, WRITE
0005	AND, 0002, WRITE
0006	OUT, 0101, WRITE
0007	LD, 0001, WRITE
0008	LD, 0003 WRITE
0009	OR, 0105, WRITE
0010	AND, LD, WRITE
0011	AND, 0004, WRITE
0012	OUT, 0102, WRITE
0013	LD, 0001, WRITE
0014	AND, 0002, WRITE
0015	AND, 0005, WRITE
0016	OUT, 0103, WRITE
0017	LD, 0001, WRITE
0018	AND, 0006, WRITE
0019	AND, 0005, WRITE
0020	OUT, 0104, WRITE
0021	FUN, 01, WRITE

After IL Is Used



Statement Logic

Address	Key Sequence
0000	LD, 001, WRITE
0001	FUN, 02, WRITE
0002	LD,NOT,0000, W
0003	AND, 0002, WRITE
0004	OUT, 0100, WRITE
0005	LD, 0002, WRITE
0006	OUT, 0101, WRITE
0007	LD, 0003, WRITE
0008	OR, 0005, WRITE
0009	AND, 0004, WRITE
0010	OUT, 0102, WRITE
0011	LD, 0002, WRITE
0012	AND, 0005, WRITE
0013	OUT, 0103, WRITE
0014	LD, 0006, WRITE
0015	AND, 0005, WRITE
0016	OUT, 0104, WRITE
0017	FUN, 03, WRITE
0018	FUN, 01, WRITE

Figure 4.14 Program Before and After Introducing An Interlock Statement

Temporary Relays (TR)

When a program cannot be formatted by using IL and ILC statements, Temporary Relays (TRs) may be used instead. Temporary Relays are program instructions used in Ladder diagrams which do not adhere to circuit or syntax rules. An appropriate application for a TR would be for a circuit consisting of several branches of output coils (Figure 4.15). This theoretical circuit would be impossible to program in its present format using 'Loader Mnemonics.' Temporary Relays are therefore needed to form sub-branches for each main rung in order to accomplish complex Logic functions and thus eliminating the need for several additional keystrokes.

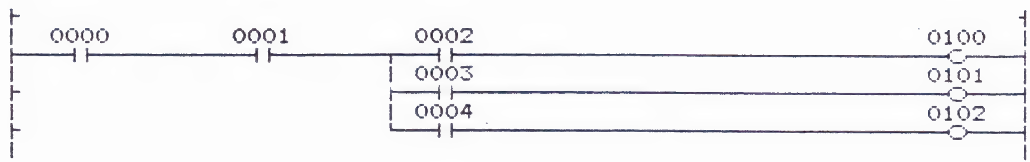


Figure 4.15 Temporary Relay Instruction

To program this example, first perform the Erase Procedure (p.76). Once the display shows 0000 indicating the starting address, follow the key sequence below.

	Address	Key Sequence
(1)	0000	LD, 0000, WRITE
(2)	0001	AND, 0001, WRITE
(3)	0002	OUT, TR, 0, WRITE
(4)	0003	AND, 0002, WRITE
(5)	0004	OUT, 0100, WRITE
(6)	0005	LD, TR, 0, WRITE
(7)	0006	AND, 0003, WRITE
(8)	0007	OUT, 0101, WRITE
(9)	0008	LD, TR, 0, WRITE
(10)	0009	AND, 0004, WRITE
(11)	0010	OUT, 0102, WRITE
(12)	0011	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

NOTE

Temporary Relays must be used with either a LD or OUT statement.

The C-Series Programmable Controllers have eight (8) Temporary Relays (TR0-TR7). All eight may be used in every rung if needed as <u>they perform no Logic statements, only Logic paths</u>. TRs however, may not be duplicated in a rung.

Temporary Relays may also be used as Master Control Relays as shown in Figure 4.16. In this example no dynamic Logic will be performed until Input 0000, the "rung permissive", is energized.

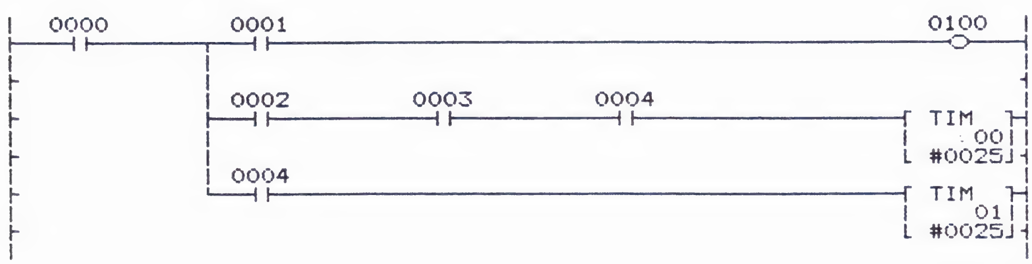


Figure 4.16 Master Control Relay

To program this example, first perform the Erase Procedure (p.76). Once the display shows 0000 indicating the starting address, press the following keys in sequence to enter the program:

	Address	Key Sequence
(1)	0000	LD, 0000, WRITE
(2)	0001	OUT, TR, 0, WRITE
(3)	0002	AND, 0001, WRITE
(4)	0003	OUT, NOT, 0100, WRITE
(5)	0004	LD, TR, 0, WRITE
(6)	0005	AND, 0002, WRITE
(7)	0006	AND, 0003, WRITE
(8)	0007	OUT, TR, 0, WRITE
(9)	0008	AND, NOT, 0004, WRITE
(10)	0009	TIM, 00,, WRITE
		# 0025, WRITE
(11)	0010	LD, TR, 0, WRITE
(12)	0011	AND, 0004, WRITE
(13)	0012	TIM, 01, WRITE
		# 0025, WRITE
(14)	0013	FUN, 01, WRITE

To verify the program, press: CLR, CLR, SRCH.

NOTES

(1) When using the Graphic Programming Console (GPC) to program a PLC with Ladder Logic statements, note that the GPC inserts Temporary Relays automatically to allow mnemonic statements to conform to the graphic Ladder diagram.

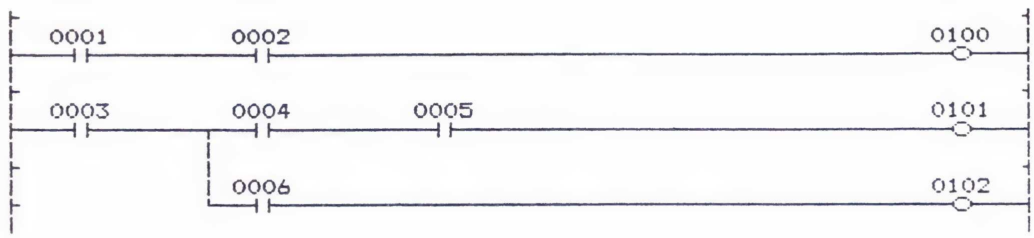
(2) There are times when it may be easier and take less memory to use TRs in place of IL statements.

Temporary Relay Placement Exercises

Fill in the statement logic for each exercise below.

Exercise No. 1

Ladder Diagram

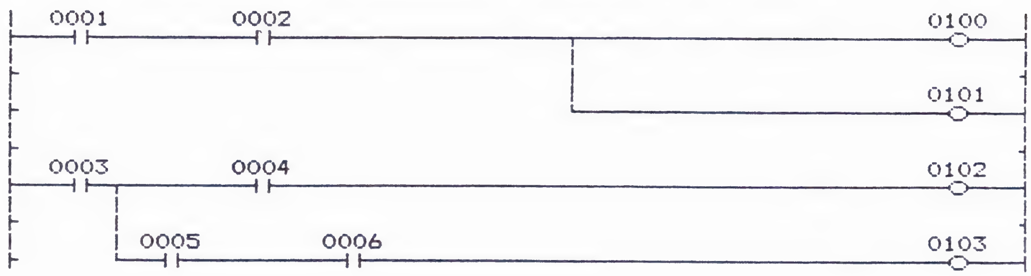


Statement Logic

Address	Key Sequence

Exercise No. 2

Ladder Diagram



Statement Logic

Address **Key Sequence**

FUN 04 and FUN 05 (Jump and Jump End)

These functions signify a Jump (JMP) and Jump End (JME) respectively. The contents of a program between these two functions are skipped or executed according to the result immediately before the JMP instruction. JMP and JME are always used in pairs.

In Figure 4.17 if Input A equals a "1" (ON) condition, the program contents between JMP and JME are scanned and updated. If however, Input A equals a "0" (OFF) condition, no I/O update will be performed.

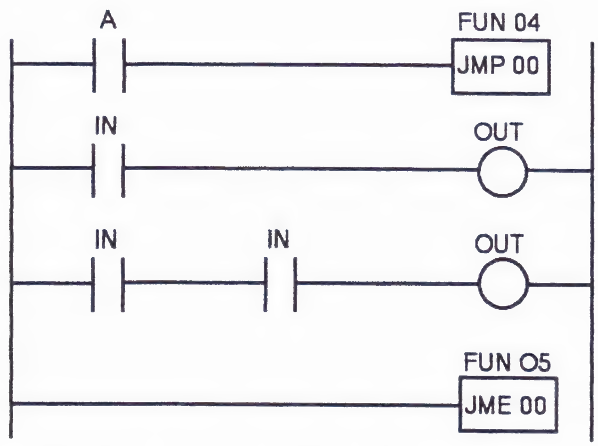


Figure 4.17 Jump and Jump End

It is important to note at this point that if logic is "Jumped" while the outputs are ON, they will remain ON until the program is scanned and updated again (not jumped).

The numerical value entered after the function number determines whether the logic between the JMP and JME statements is scanned. If the numerical value is 00, then the logic between the two statements is scanned. However, if the numerical value is between 01-08, the logic between the JMP and JME statements is not scanned.

As long as the two statements are used in pairs, an unlimited number of JMP 00/JME 00 may be used since their boundries are called out by their pairing. On a JMP statement with numerical data (JMP 01), there is a limit of 8 separate JMP statements.

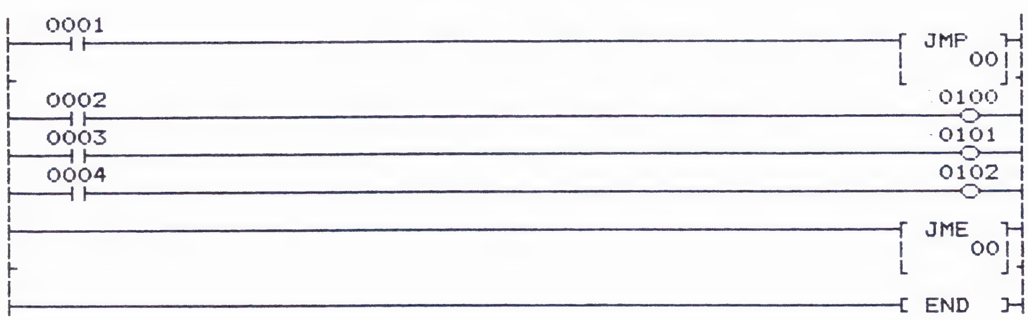
It is important to understand what the PLC does when a JMP 00 or JMP 01-08 is addressed. Referring back to Figure 4.17, if a JMP 00 is used, the PLC is commanded to jump a group of logic (between JMP & JME) and will bypass the I/O update portion of logic which is part of the scan. The rest of the program is still scanned.

If however JMP 01-08 and JME 01-08 statements are used, the PLC will not scan any logic between the two statements resulting in a shorter scan time.

A few programming examples using JMP/JME statements are provided below.

Example No. 1

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	FUN, 04, 00, WRITE
0002	LD, 0002, WRITE
0003	OUT, 0100, WRITE
0004	LD, 0003, WRITE
0005	OUT, 0101, WRITE
0006	LD, 0004, WRITE
0007	OUT, 0102, WRITE
0008	FUN, 05, 00, WRITE
0009	FUN, 01, WRITE

Verify by pressing: CLR, CLR, SRCH.

Once the program has been entered and verified, switch the Programming Console to the MONITOR mode. (The PLC will not perform Logic in the MONITOR mode.) Perform the following steps on the I/O Simulator:

(1) With Input 0001 OFF, energize Inputs 0002, 0003, and 0004. Note that no outputs are energized.

(2) Turn OFF all the inputs (0002, 0003, 0004).

(3) Energize Input 0001. Then energize Inputs 0002, 0003, and 0004. Note that all of the logic is performed and Outputs 0100, 0101, and 0102 energize.

(4) Turn OFF Input 0001 (JMP permissive). Then turn OFF Inputs 0002, 0003, and 0004.

    Note that the outputs remain ON because the program was jumped before Inputs 0002, 0003, and 0004 were turned OFF.

(5) To perform the program logic again, turn ON Input 0001. Outputs 0100, 0101, and 0102 de-energize because the PLC performs the logic and sees Inputs 0002, 0003, and 0004 as OFF.

As you can see from this procedure, it is possible to lock logic ON or OFF by using a JMP/JME statement.

Example No. 2

Jump statements are also used in programs that contain multiple coil outputs (Figure 4.18)

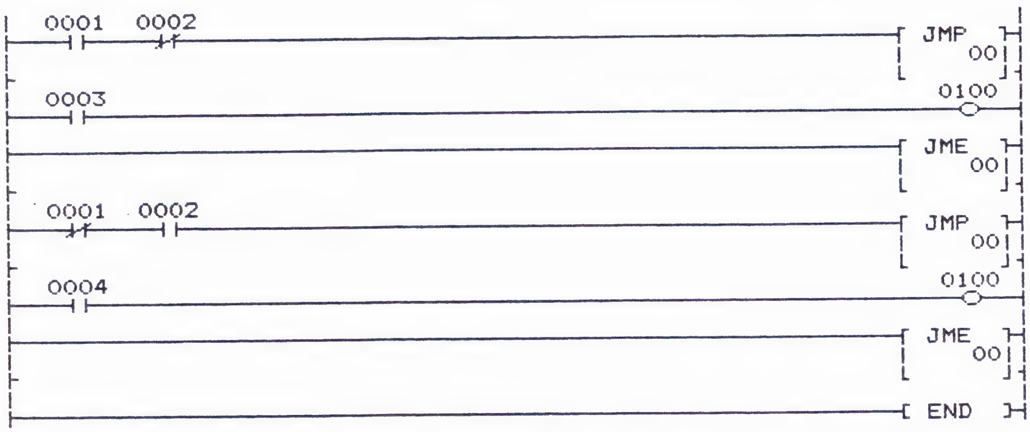


Figure 4.18 JMP Used With Multiple Coil Outputs

Note that:

> If Input 0001 AND NOT Input 0002 are energized, then Input 0003 will turn on Output 0100.
>
> If NOT Input 0001 AND Input 0002 are energized, then Input 0004 will turn on Output 0100.

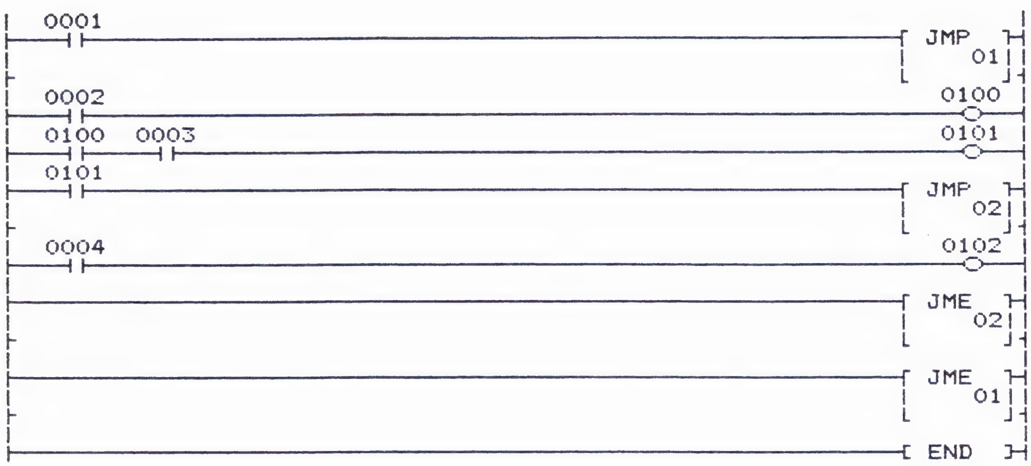
This is known as "Exclusive Logic". This may be likened to a motor system in that you can run a motor forward or in reverse, but you cannot run a motor forward and reverse at the same time.

Example No. 3

As noted previously, when JMP and JME statements are used with numerical data, i.e. JMP 01/JME 01, JMP 02/JME 02, etc., program processing time is saved because the program between FUN 04 and FUN 05 will not be scanned or updated.

By using JMP/JME with numerical data, it is also possible to "Nest" Jump statements (i.e., placing one Jump statement within another Jump statement) as shown in Figure 4.19.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	FUN, 04, 01, WRITE
0002	LD, 0002, WRITE
0003	OUT, 0100, WRITE
0004	LD, 0100, WRITE
0005	AND, 0003, WRITE
0006	OUT, 0101, WRITE
0007	LD, 0101, WRITE
0008	FUN, 04, 02, WRITE
0009	LD, 0004, WRITE
0010	OUT, 0102, WRITE
0011	FUN, 05, 02, WRITE
0012	FUN, 05, 01, WRITE
0013	FUN, 01, WRITE

Verify by pressing: CLR, CLR, SRCH.

Figure 4.19 Nesting Jump Statements

After entering and verifying the program, switch the Programming Console to the MONITOR mode and use the I/O simulator to perform the following:

(1) Energize Input 0001. This allows the PLC to scan between JMP 01 and JME 01.

(2) Energize Inputs 0002 and 0003. This allows the PLC to scan between JMP 02 and JME 02.

FUN 11 (Keep Relay)

This function is used to indicate a Keep Relay (also called a Latching Relay). FUN 11 uses two control lines: Set and Reset. These two lines toggle the relay ON (Set) or OFF (Reset) and must be enabled alternately. Because the Keep Relay is an instruction, any I/O relay including Real I/O and Auxiliary I/O may be used.

FUN 11 may be used to simulate Start/Stop, Latching, and Stop circuits.

Figure 4.20 shows a conventional Ladder latch.

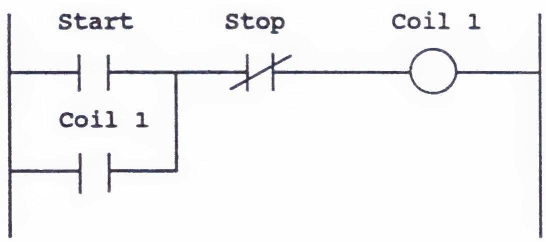


Figure 4.20 Conventional Latch

If the Start input equals a "1" (ON) condition, Output Coil 1 will energize and remain energized even if the Start input turns OFF (latches itself on by Coil 1). Coil 1 will remain ON until the Stop input is energized, breaking the circuit.

Figure 4.21 shows a latch using the FUN 11 function.

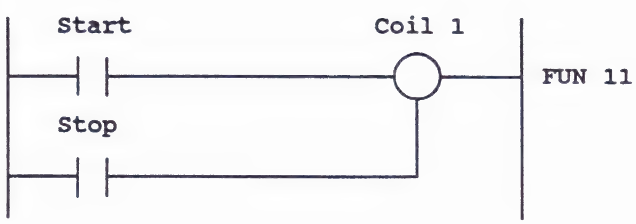
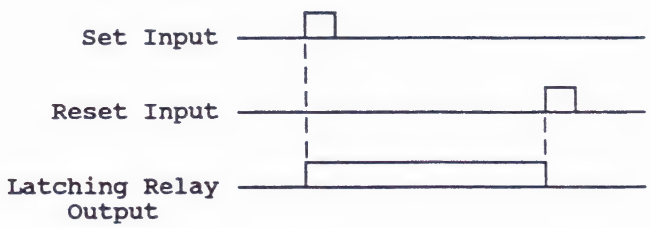


Figure 4.21 Keep Relay

The Start input toggles Coil 1 ON, while the Stop input toggles Coil 1 OFF.

By using a Holding Relay (HR) location, the Latching circuit can become memory retentive. Should a power failure occur, uninterrupted machine control is assured. Figure 4.22 shows a timing chart and an example of a program containing a Keep Relay in an HR location.

Timing Chart



Ladder Diagram

Figure 4.22 Keep Relay (FUN 11)

To program this example, first erase the previous program by performing the Erase Procedure (p.76). Then follow the key sequence below to enter the program:

Address	Key Sequence
0000	LD, 0001, WRITE (Set)
0001	LD, 0002, WRITE (Reset)
0002	FUN, 11, HR 000, WRITE (Keep)
0003	LD, HR 000, WRITE
0004	OUT, 0100, WRITE
0005	FUN, 01, WRITE

Verify by pressing: CLR, CLR, SRCH.

NOTE

There is no limit to the amount of FUN 11 instructions you may use in a single program.

FUN 13 and FUN 14 (Differentiation Up and Down)

These functions indicate Differentiation Up (DIFU) and Differentiation Down (DIFD) respectively, and are more commonly known as transitional contacts.

A DIFU (FUN 13) will turn ON a specified coil for one (1) scan time when the input to this "DIF" statement goes from an OFF to an ON condition.

A DIFD (FUN 14) will turn ON a specified coil for 1 scan time when the input to this DIF statement goes from an ON to an OFF condition.

Both functions therefore supply a signal to the PLC only upon input transition - the DIFU at the leading edge of the input and the DIFD at the trailing edge of the input (see Figure 4.23).

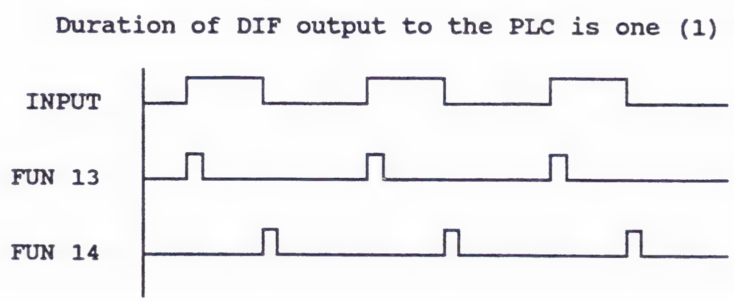


Figure 4.23 Timing Chart for DIFU and DIFD

NOTES

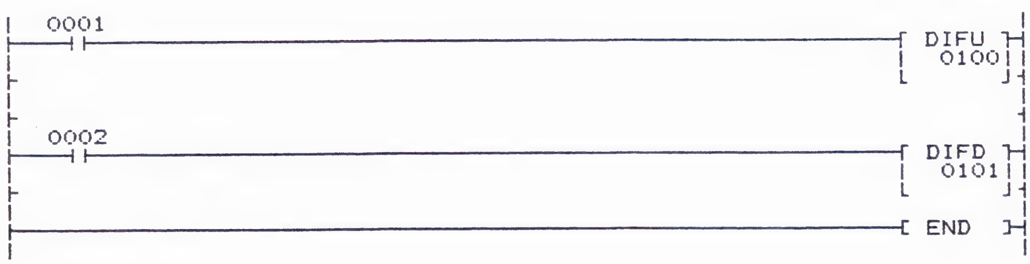
(1)	The minimum amount of time for the PLC to accept any information as valid is one scan time.
(2)	A maximum of 48 DIF statements (DIFU & DIFD) may be programmed. If this is exceeded, the 49th DIF will be treated as NOP and the PLC will display the error message "DIF OVER" on the Programming Console.

DIFU and DIFD may be used with the following:

I/O, Internal Auxiliary Relays	0000 to 1807
Holding Relays	000 to 915

Figure 4.24 shows a sample program using FUN 13 and 14.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	FUN, 13, 100, WRITE (DIFU)
0002	LD, 0002, WRITE
0003	FUN, 14, 101, WRITE (DIFD)
0004	FUN, 01, WRITE

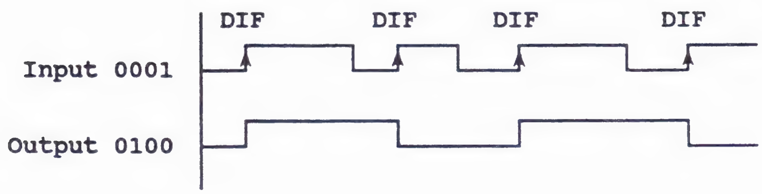
Figure 4.24 FUN 13 and FUN 14

In this case, the example above is not a good working program as the output would only be on for 1 scan - not long enough to drive a "real world" device.

DIF statements are best utilized in processes that require an action to take place only when an input goes from low to high (OFF to ON) or high to low (ON to OFF).

Figure 4.25 shows a Start/Stop circuit using one pushbutton. This type of a circuit is also known as a Flip-flop, Alternating, Binary Counter, and Push-on/Push-off circuit.

Timing Chart



Ladder Diagram

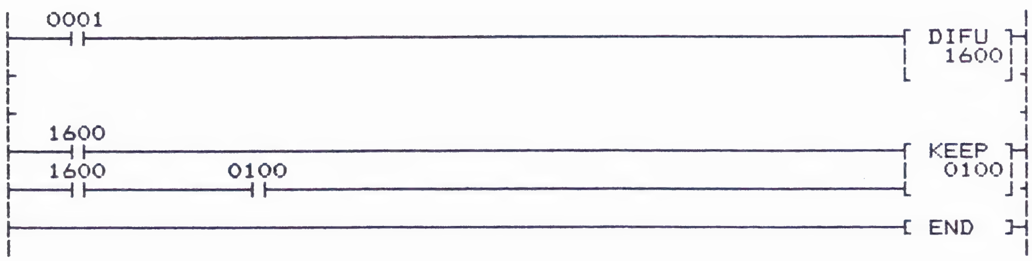


Figure 4.25 Start/Stop Circuit

Output 0100 makes a transition to ON-OFF/OFF-ON whenever Input 0001 goes from an OFF-ON transition. To program this example, clear the previous program (p.76) and enter the key sequence below:

Address	Key Sequence
0000	LD, 0001, WRITE
0001	FUN, 13, 1600, WRITE (DIFU)
0002	LD, 1600, WRITE
0003	LD, 1600, WRITE
0004	AND, 0100, WRITE
0005	FUN, 11, 0100, WRITE (Keep)
0006	FUN, 01, WRITE

Verify by pressing: CLR, CLR, SRCH.

Now run the program as follows:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press the CLR key to clear the display.

(3) Monitor the I/O by pressing:

    (a) OUT, 0100, MONTR
    (b) OUT, 0001, MONTR

Note the transition of Output 0100 everytime Input 0001 is activated.

NOTE

Many of the special functions (such as Math, Shift, etc.) require DIF statements to assure a single execution of the function statement.

Applicaton Example

DIF Statements are effectively used with a sensor on a material handling system (conveyor) to guarantee quick system response (see Figure 4.26 below).

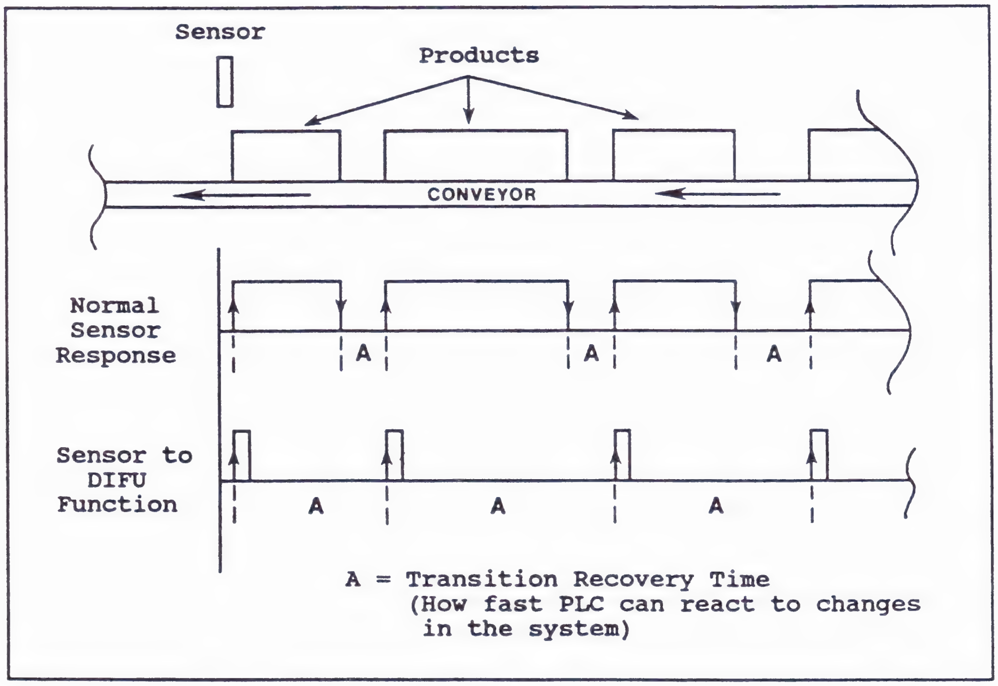


Figure 4.26 Conveyor System Using DIF Statements

Notice the shorter recovery time in the second timing chart when a sensor is used with a DIFU function.

As you get into more extensive programming, you will likely use DIF Statements many times.

Timers (TIM) and Counters (CNT) - An Overview

In addition to standard relay logic such as LD, AND, OR, NOT, etc., Timers (TIMs) and Counters (CNTs) are the second oldest pair of PLC instructions. While many early styles of PLCs did not include timer and counter instructions, all PLCs on the market today offer at least one type of timer instruction.

Uses for timers can be as simple as providing a start-up warning delay in a system, or as complex as providing timing steps for batch processing.

The C20K Family of PLCs has 48 separate registers (00-47) which may be programmed as timers or counters. It is important to note at this time that because both timers and counters share the same registers, you must be careful not to duplicate addresses. For example, if you have a TIM 00 you cannot have CNT 00. The same duplication error applies to timers and counters as it does to output coils.

Also note that timers are reset should a power failure occur, but counters are not.

Timers

Note the following important facts about timers:

(a) Each timer can be set for a minimum of 0.1 second through a maximum time range of 999.9 seconds, in increments of 0.1 second.

(b) Timers are of a decrementing type producing an output when the present value (time remaining) becomes 0000. When the timer input is de-energized (turned off), the present value of the timer returns to the preset value.

(c) A timer is programmed like an internal relay except that the OUT instruction does not precede the timer number. The time delay can be entered at the same time and at the same address as the timer coil. A 4-digit value which is 10 times the actual time in seconds, is used to set the timer. For example, a delay of 7 seconds would be 070 , a delay of 25 seconds would be 250 , etc.

(d) Timers have an ON-delay action meaning that timing starts only when a coil is energized. In other words after an operator-specified delay time, the contacts energize and remain energized for as long as the coil is kept energized.

Though there are several ways to implement a time proccess in a machine control operation, one of the easiest ways to illustrate this is by creating a timing or logic chart for a Ladder diagram (see Figure 4.27).

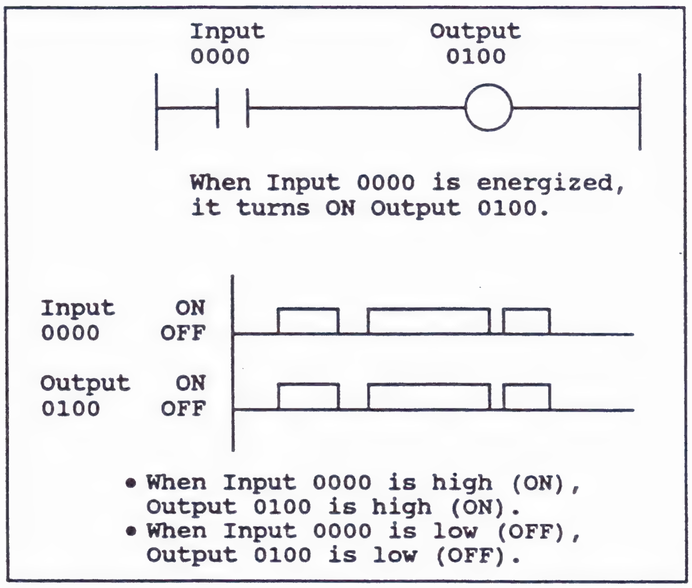
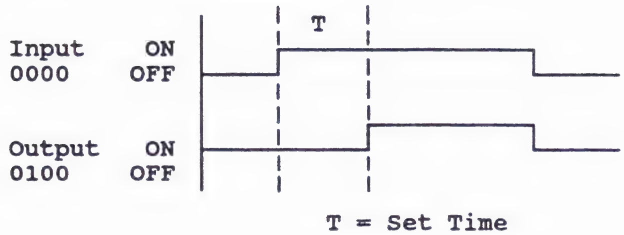


Figure 4.27 Timing Chart

Though there is no obvious timing function in this chart, it serves to show the relationship between the ON and OFF state of Input 0000 and the ON and OFF state of Output 0100.

By addressing a time delay function of 2.5 seconds after Input 0000 goes high (ON), we can delay energizing Output 0100. This is called a "Delay-On" or "On-Delay" function. Because the delay is started with an input in this case, we will call this a Signal On-Delay function (see time chart).

SIGNAL ON-DELAY TIMER



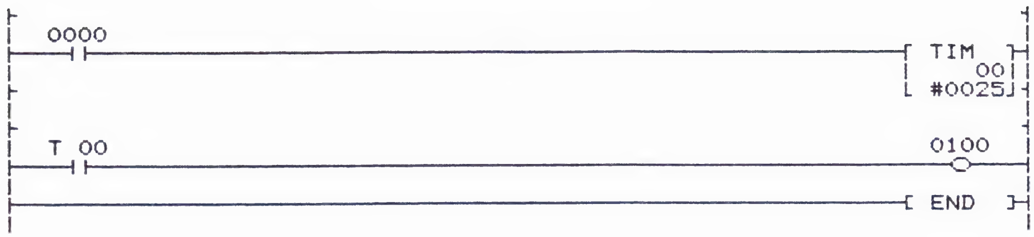
Note that the "SET TIME" is a value programmed by the operator. Once again, when Input 0000 is energized, 2.5 seconds (Set Time) elapse before Output 0100 is energized. When the input signal goes low (OFF) the output will immediately turn OFF.

NOTE

It is important to note at this point that Omron timers are non-retentive. For example, if Input 0000 (in our example) went OFF before the time cycle was complete, i.e., the timer timed out and energized the output, the timer would reset to its preset value (in this case 2.5 seconds).

Figure 4.28 shows the Ladder diagram and Statement Logic to enter our program example.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0000, WRITE
0001	TIM, 00, WRITE
	# 0025, WRITE
0002	LD, TIM, 00, WRITE
0003	OUT, 0100, WRITE
0004	FUN, 01, WRITE

Verify by pressing: CLR, CLR, SRCH.

Figure 4.28 Signal On Delay Timer

To run this program, follow the steps below:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press the CLR key to clear the display.

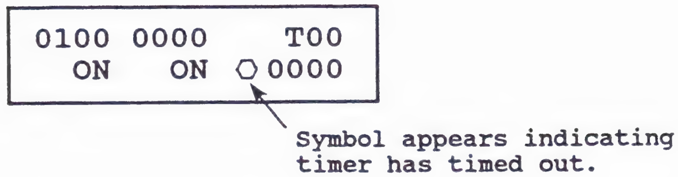
(3) Monitor the I/O by pressing:

(a) TIM, 00, MONTR
(b) OUT, 0000, MONTR
(c) OUT, 0100, MONTR

Display will show:

```
0100 0000    T00
 OFF  OFF   0025
```

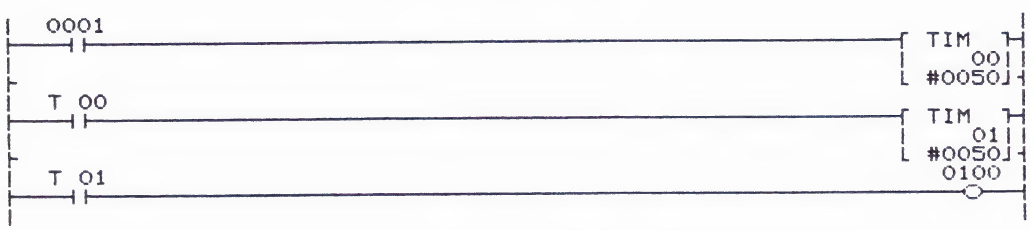
(4) Energize Input 0000 on the I/O Simulator. Note that the timer times down. The display shows:



(5) Reset the inputs and try some variations. For example, leave Input 0000 on for less than 2.5 seconds and notice how the timer automatically resets (because its non-retentive).

If a control operation requires a time delay greater than 999.9 seconds, two or more timers may be cascaded together so that at 9999 seconds the first timer starts the next timer, and so forth. Figure 4.29 shows an example of Cascading Timers.

Ladder Diagram



Statement Logic

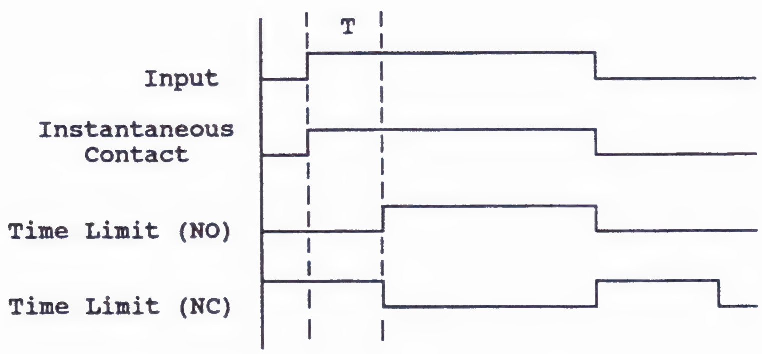
Address	Key Sequence
0000	LD, 0001, WRITE
0001	TIM, 00, WRITE
	# 0050, WRITE
0002	LD, TIM, 00, WRITE
0003	TIM, 01, WRITE
	# 0050, WRITE
0004	LD, TIM, 01, WRITE
0005	OUT, 0100, WRITE
0006	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.28 Cascading Timers

Note that both Timers 00 and 01 are set for 5 seconds. Therefore, Output 0100 will energize 10 seconds after Input 0001 is energized.

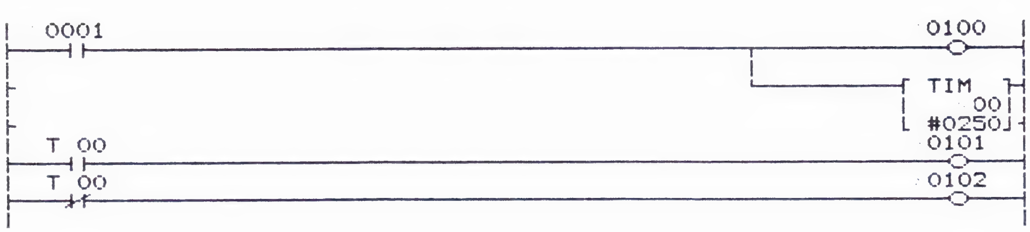
There are instances when the operator will require a timing instruction that offers instantaneous output contacts. An Instantaneous Contact is a Normally Open or Normally Closed contact which is activated when a timer receives an input signal. In contrast, timed contacts (NO or NC) are activated only after a timer's preset time delay has expired. The time chart below depicts both Time Limit Contacts and Instantaneous Contacts.



T = Set Time

Figure 4.30 shows a Ladder Diagram and Statement Logic for a program example that utilizes both Time Limit and Instantaneous Contacts.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	OUT, 0100, WRITE
0002	TIM, 00, WRITE
	# 0250, WRITE
0003	LD, TIM, 00, WRITE
0004	OUT, 0101, WRITE
0005	LD, NOT, TIM, 00, WRITE
0006	OUT, 0102, WRITE
0007	FUN, 01, WRITE

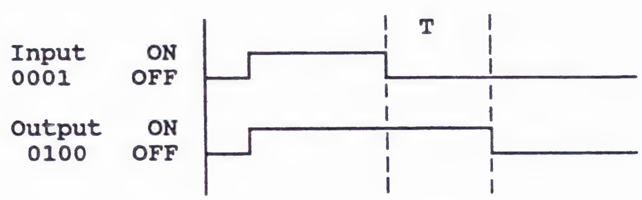
To verify press: CLR, CLR, SRCH.

Figure 4.30 Time Limit and Instantaneous Timing Functions

Because of all the logic available in the PLC, the operator can configure many types of timing modes. The following paragraphs describe three examples including Signal Off-Delay, One Shot, and Repeat Cycle Timers.

Signal Off-Delay Timer

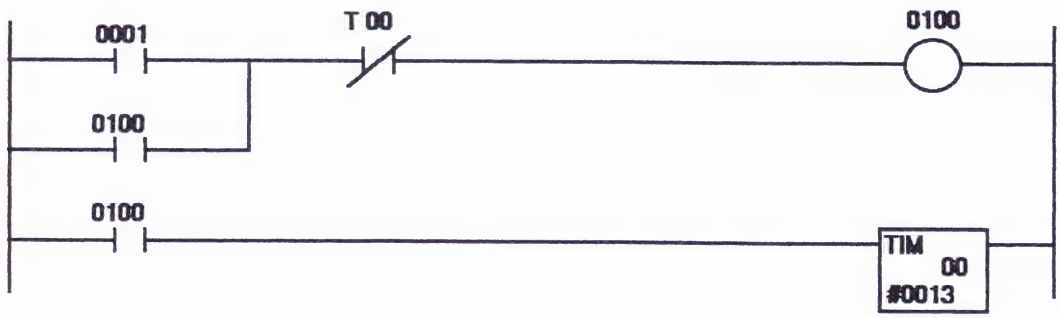
The timing chart below illustrates a Signal Off-Delay Timer. An automatic garage door opener utilizes this type of timing mode. When the door is activated, the light in the garage turns ON. When the door is closed, the light remains ON for a specified time and then turns OFF.



T = 1.3 seconds

When Input 0001 goes high (ON), Output 0001 goes high (ON) and will remain ON until Input 0001 is low (OFF) for at least 1.3 seconds. Figure 4.31 depicts the Ladder Diagram and Statement Logic for a Signal Off-Delay Timer.

Ladder diagram



Statement Logic

Address		Key Sequence
0000	---	LD, 0001, WRITE
0001	---	OR, 0100, WRITE
0002	---	AND, NOT, TIM, 00, WRITE
0003	---	OUT, 0100
0004	---	LD, 0100, WRITE
0005	---	TIM, 00, WRITE
		#, 0013, WRITE
0006	---	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

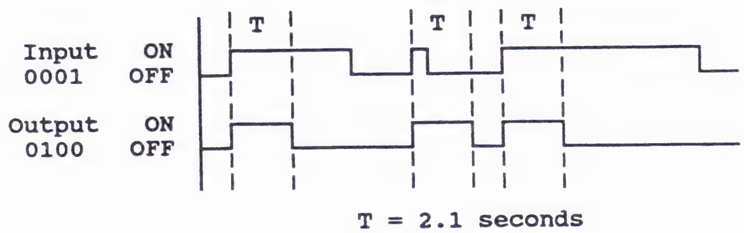
Figure 4.31 Signal Off-Delay Timer

Note that both the Signal On and Signal Off modes condition the input signal. In the On-delay mode, a signal must be present for at least the set time of the timer before the logic will accept it as a valid signal. In the Off-delay mode, the signal must be absent (only after being present) for the set time before the logic will accept it as a valid signal and turn off the output.

The next two timer examples condition the signal of the output in terms of duration, etc.

One Shot Timer

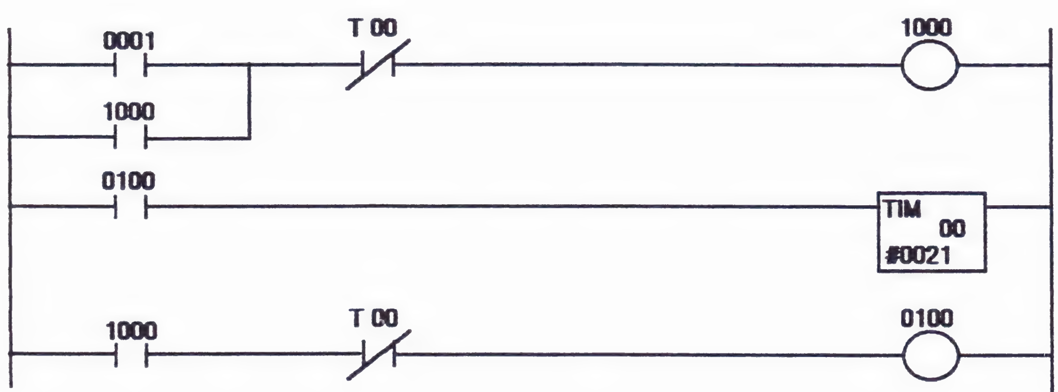
Everytime an input is energized, an output is energized for a predetermined time in a One Shot timing instruction. The chart below shows a timing diagram for a One Shot Timer.



The timing chart shows that Output 0100 is energized for 2.1 seconds regardless of the duration of Input 0001.

Figure 4.32 depicts the Ladder Diagram and Statement Logic for a sample One Shot timer. To enter this program erase the previous program (p.76) and follow the Key Sequence in order.

Ladder Diagram



<u>Statement Logic</u>

Address	Key Sequence
0000	LD, 0001, WRITE
0001	OR, 1000, WRITE
0002	AND, NOT, TIM, 00, WRITE
0003	OUT, 1000, WRITE
0004	LD, 0100, WRITE
0005	TIM, 00, WRITE
	#, 0021, WRITE
0006	LD, 1000, WRITE
0007	AND, NOT, TIM, 00, WRITE
0008	OUT, 0100, WRITE
0009	FUN, 01, WRITE

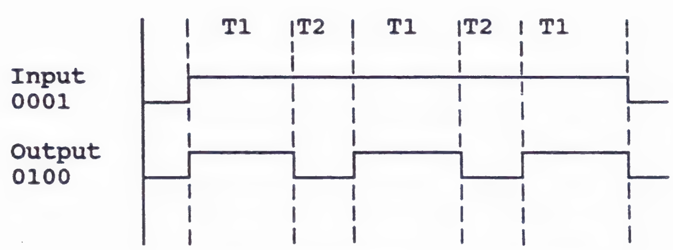
To verify press: CLR, CLR, SRCH.

Figure 4.32 One Shot Timer

Many times the PLC may execute program logic so quickly that there may be problems turning on large solenoids that have a slow "pick time". In this case, a One Shot Timer would alleviate this type of difficulty when properly used in a program.

Repeat-Cycle Time

The timing chart below depicts a Repeat-Cycle Timer. This type of timing application is used to separate ON and OFF times by "cycling" an output device.

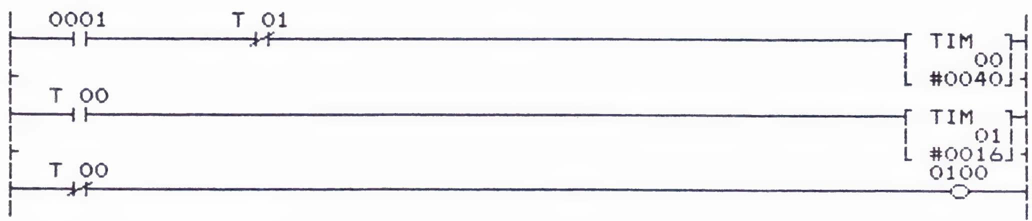


T1 = ON time: 4.0 seconds
T2 = OFF time: 1.6 seconds

This timing diagram graphically depicts the response of the repeat cycle. After the timing cycle permissive (Input 0001) is actuated, Output 0100 will energize for 4.0 seconds, turn OFF for 1.6 seconds, and repeat this cycle as long as Input 0001 remains ON.

Figure 4.33 shows the Ladder Diagram and Statement Logic for a Repeat-Cycle Timer. To enter the program first erase the previous program (p.76) and follow the Key Sequence.

<u>Ladder Diagram</u>



<u>Statement Logic</u>

Address	Key Sequence
0000	LD, 0001, WRITE
0001	AND, NOT, TIM, 01, WRITE
0002	TIM, 00, WRITE
	# 0040, WRITE
0003	LD, TIM, 00, WRITE
0004	TIM, 01, WRITE
	# 0016, WRITE
0005	LD, NOT, TIM, 00, WRITE
0006	OUT, 0100, WRITE
0007	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.33 Repeat-Cycle Timer

A Repeat-Cycle Timer is typically used when a specified ON and OFF time is required in a control cycle. For example, the administration of cutting oil to a metal working process (lathes, drills, etc.) would require a Repeat-Cycle Timer.

NOTE

Because the timer is set up using Ladder Logic, the "OFF" time can precede the "ON" time in the control timing process. For example, you can substitute the instruction "LD, TIM, 00, WRITE" at Address 0005 in place of "LD, NOT, TIM, 00, WRITE".

As you can see, there are numerous applications for timers in PLC control and these timers allow the operator a great amount of flexibility, and the same precision offered by digital stand-alone timers.

FUN 15 (High-Speed Timer)

By using the FUN 15 (TIMH) instruction instead of the TIM key, the timing cycle is changed in terms of time unit measurements. TIMH measures in units of 0.01 second from a minimum time cycle of 0.01 seconds to a maximum of 99.99 seconds.

NOTE

Only the first 16 timers (00-15) can be used as High-Speed Timers.

Changing Timer Values

The following procedures explain how to change preset and present timer values, as well as forcing a timer.

Preset Value

This operation may be performed in the PROGRAM or MONITOR modes while the program is being executed. Refer back to Figure 4.33 and perform the following steps:

(1) Clear the display by pressing: CLR, CLR.

(2) Search for the timer by pressing: TIM, 00, SRCH.
This brings you to Address 0002 where TIM 00 is located.

(3) Press the Down Arrow key. This brings you to the preset area: #0040.

(4) Press the CHG key to change the value.

(5) Enter the new value: 0035, WRITE.
The new value is now recorded.

(6) To call up the new preset value press:

CLR, CLR, TIM, 00, MONTR

Present Value

The "present value" of a timer refers to the current value from which the timer is counting down during a program execution. The present value can only be changed for one time cycle. When the timer is reset (manually or through program logic), the timer will revert back to the preset value.

To understand how this works, we will use the program in Figure 4.30 as our example. Our task will be to change the value of Timer 00 while it is in its timing process. First enter and verify the program. Then, perform the following steps:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press the CLR key to clear the display.

(3) Press: TIM, 00, MONTR.

(4) Energize Input 0001 on the I/O simulator to start the timing process. The timer should begin decrementing.

(5) Press the CHG key. The display should show:

```
0000PRES VAL?
    T00 0250????
```

(6) At this point, we will add 10 seconds to Timer 00. Press: 0350, WRITE.

(7) The timer will begin a new timing cycle decrementing from 35 seconds.

(8) After the timer has timed out or the timer has been manually reset, it will revert back to the original preset time of 0250 (25 seconds).

Forcing a Timer

The operator may also force a timer into a different state. Perform the following steps in sequence:

(1) Make sure the Programming Console is placed in MONITOR mode.

(2) Press the CLR key twice to obtain Address 0000.

(3) Press: TIM, 00, MONTR.

(4) Start the timing process by energizing Input 0001 on the I/O simulator. The timer should begin timing down.

(5) By pressing the REC/RESET key, the timer will RESET to the preset value and start the timing cycle again.

(6) By pressing the PLAY/SET key, the timer contact will be forced ON setting the timer to 0000. The display should show:

Symbol appears indicating the timer has timed out.

```
T00
○0000
```

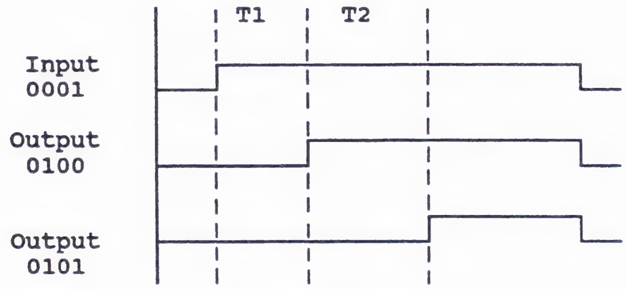
NOTE

> Only preset timer values are memory retentive. If the PLC loses power or the unit is switched to the PROGRAM mode, the timer will RESET back to the preset value.

Timer Exercises

Two exercises are provided below to reinforce the timer information presented in the previous paragraphs.

Exercise No. 1



T1 = 1.8 seconds
T1 and T2 = 5.7 seconds

This diagram shows a timing process for a program. Provide answers for the following:

(1) What is the duration of T2?

(2) Draw a Ladder Diagram for the Timing Chart depicted above.

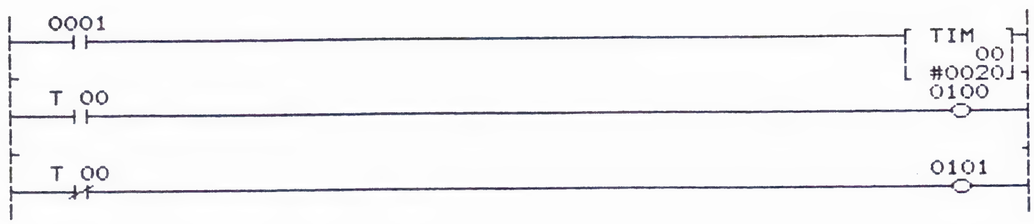
(3) Work out the Statement Logic for the Ladder Diagram and then enter the program into the PLC. Debug and run the program utilizing the I/O simulator.

Statement Logic

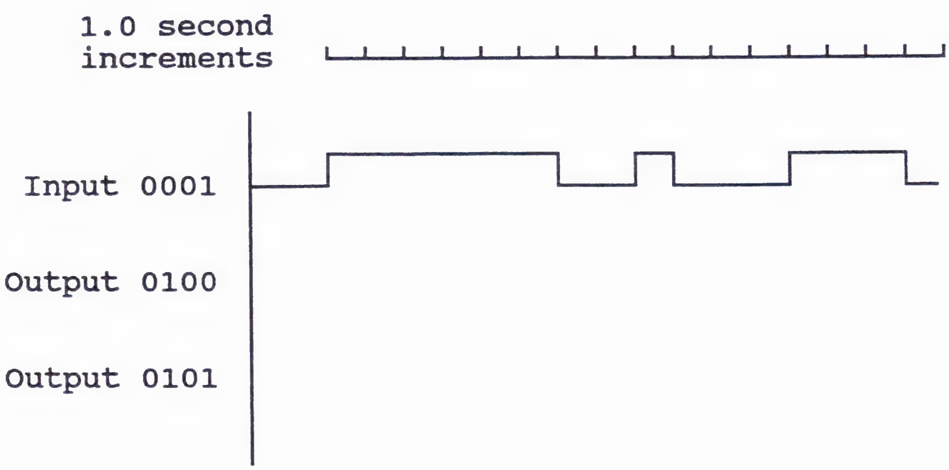
Address	Key Sequence

Exercise No. 2

Ladder Diagram



Complete the Timing Chart below:



Counters

The C20K Family contains 48 counters with address identification numbers of CNT 00-47. Remember that counters share the same register with timers and therefore numbers cannot be duplicated. Count values can be preset from 0000 to 9999. Both Normally Open (NO) and Normally Closed (NC) contacts can be used with each counter in the required quantity.

Omron counters are of the decrementing type meaning they start their countdown at a preset value and energize an output when the present value reaches 0000. The counter counts the occurrence of an event (its count input), and decrements the preset value by one (1) on the leading edge of its input.

The present value of a counter returns to its preset value (programmed by the user) and resets its counter contact only when a reset signal is applied, preventing further counting until the next specified user cycle. Counters are therefore programmed in the order of:

(a) Count input
(b) Reset input
(c) Set value
(d) Counter coil

There are three primary counting modes available:

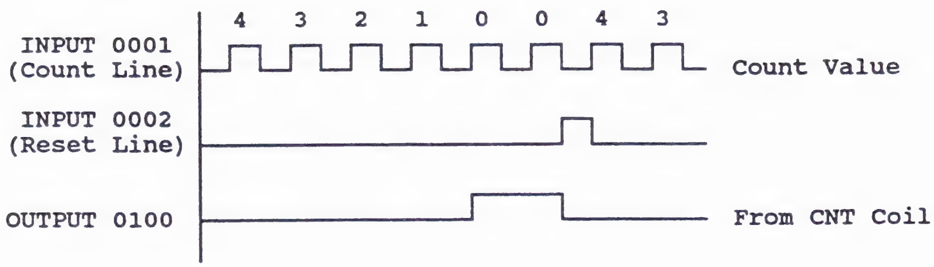
(1) Preset Counter - counts down from a specific preset count and the output energizes at 0000.

(2) Totalizing Counter - counts the total number of products run in increments. There is no output involved.

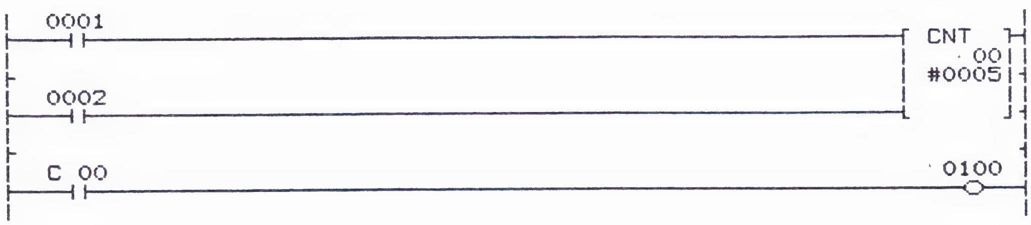
(3) Ring Counter - A Reversible Counter in which the output will energize only when the present value of the counter passes 0000.

We are going to focus primarily on the Preset Counter. Figure 4.34 shows an example of a Preset Counter complete with Ladder Diagram, Timing Chart, and Statement Logic.

Timing Chart



Ladder Diagram



NOTES

(1) PLC will not continue to count until a Reset input is applied.

(2) If a Count input and a Reset input are applied simultaneously, the Reset input takes precedence over the Count input.

(3) Because the Count and Reset lines contain two separate types of logic, each line uses a Load (LD) statement. This allows the operator to use multiple contacts for AND, OR, etc., statements used in separate lines of logic.

Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	LD, 0002, WRITE
0002	CNT, 00, WRITE
	# 0005, WRITE
0003	LD, CNT, 00, WRITE
0004	OUT, 0100, WRITE
0005	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.34 Preset Counter

To observe the operation of the Preset Timer, perform the following steps after entering and verifying the program:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press the CLR key to clear the display.

(3) Monitor Counter 00, Contacts 0001, 0002, and Output 0100. Since the display will show the status of only 3 contacts, monitor Output 0100 on the LED indicator on the CPU.

The keystrokes are as follows:

(a) CNT, 00, MONTR
(b) SHIFT, CONT/#, 0001, MONTR
(c) SHIFT, CONT/#, 0002, MONTR

The display should show:

```
0002 0001    C00
 OFF  OFF   0005
```

(4) Toggle Input 0001 ON and OFF on the I/O simulator. Notice the count value decrementing. As with timers, when a counter has "counted out" (reached value 0000), Output 0100 will energize and the octagon symbol will appear in the lower left hand corner of the display.

NOTES

(1)	Output 0100 will remain energized until it is reset by Input 0002.
(2)	After the counter has "counted out" it will not accept any input pulses until it is reset by Input 0002.

Cascading Counter

A count value greater than 9999 may be achieved by cascading two or more counters as shown in Figure 4.35 below.

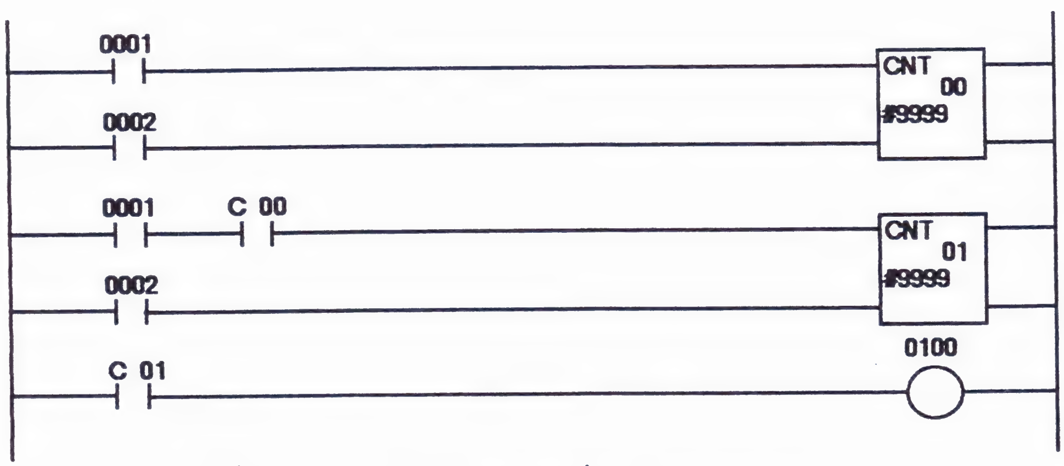


Figure 4.35 Cascading Counters

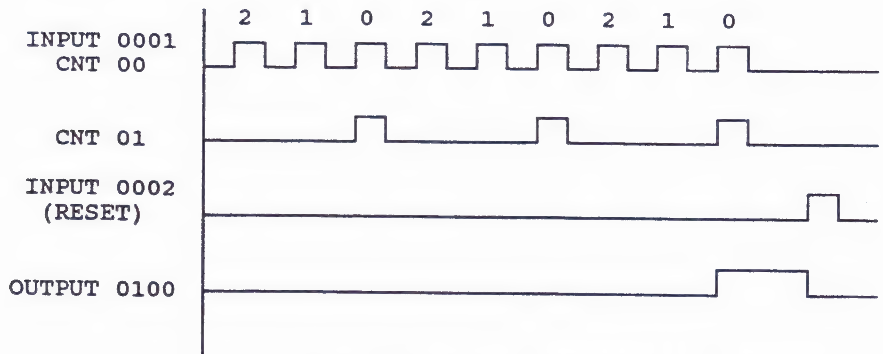
In this example, 19,998 counts will elapse before Output 0100 will energize. If three (3) counters were cascaded, a total of 29,997 would elapse before Output 0100 would energize.

Prescaling Counter

By using the same type of logic used in a Cascading Counter (see Figure 4.35 above), the operator can create a Prescaling style counter. This type of counter operation prescales the counts of one (1) input to equal a prescribed amount of counts for a second counter. For example, 10 counts of Counter 00 equals 1 count of Counter 01.

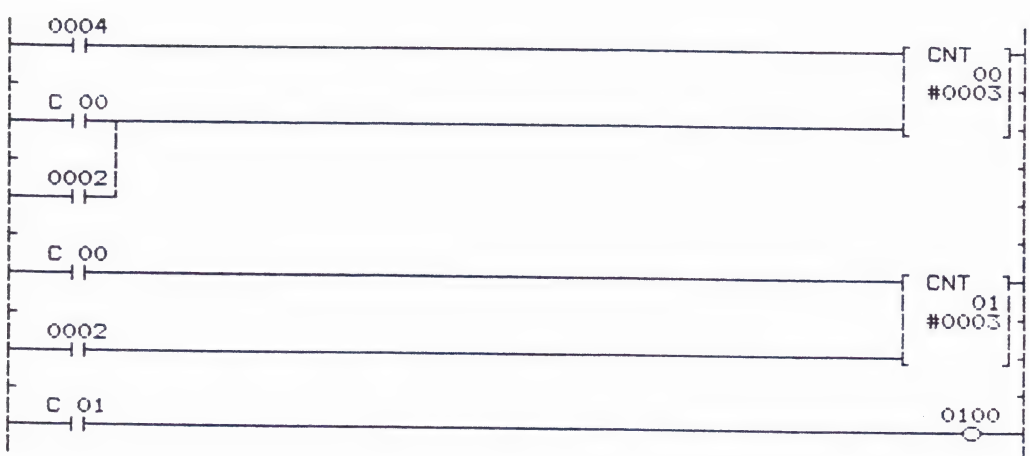
Figure 4.36 shows a Timing Chart, the Ladder Diagram, and Statement Logic for a Prescaling Counter.

Timing Chart



* Nine (9) actual counts will elapse before Output 0100 energizes.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	LD, CNT, 00, WRITE
0002	OR, 0002, WRITE
0003	CNT, 00, WRITE
	# 0003, WRITE
0004	LD, CNT, 00, WRITE
0005	LD, 0002, WRITE
0006	CNT, 01, WRITE
	# 0003, WRITE
0007	LD, CNT, 01, WRITE
0008	OUT, 0100, WRITE
0009	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.36 Prescaling Counter

In the Statement Logic for this example, the duration of the counter coil of CNT 00 will be one (1) scan (long enough for the PLC to accept this as a valid internal input).

After entering and verifying the program in Figure 4.36, monitor the program as follows:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press the CLR key to clear the display.

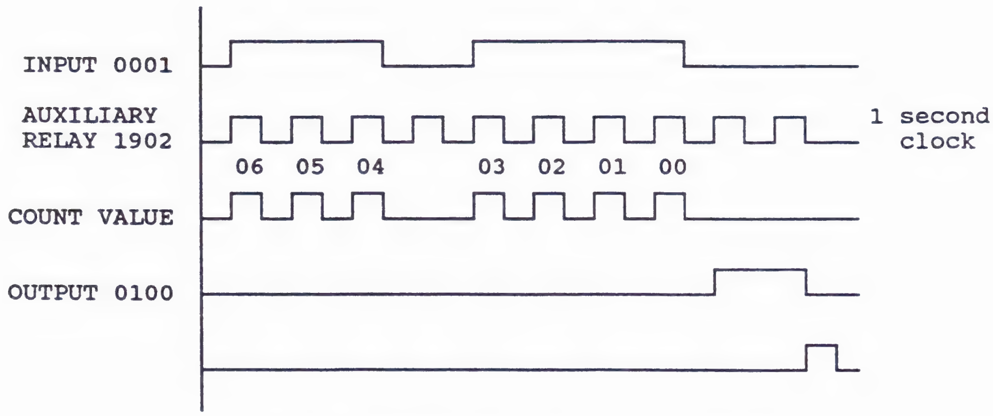
(3) Monitor the I/O by pressing:

(a) SHIFT, CONT/#, 0001, MONTR
(b) CNT, 00, MONTR
(c) CNT, 01, MONTR

Counter Used As A Retentive Timer

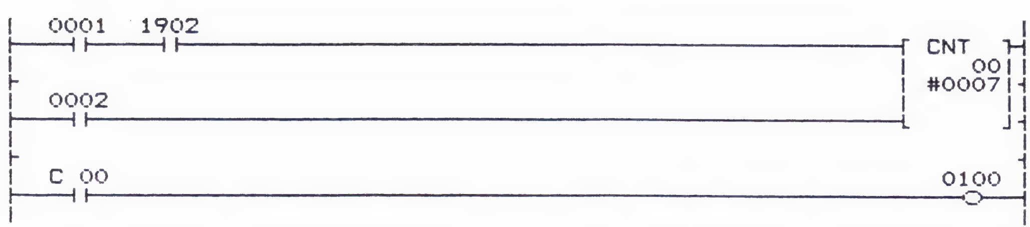
Because timers do not retain their status and will be reset if a power failure occurs, a counter may be programmed to serve as a Retentive Timer as shown in Figure 4.37. Note that Output 0100 will energize after the total accummulated ON time of Input 0001 reaches 7 seconds.

Timing Chart



Auxiliary Relay 1900 = 0.1 second
" " 1901 = 0.2 second
" " 1902 = 1.0 second

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	AND, 1902, WRITE
0002	LD, 0002, WRITE
0003	CNT, 00, WRITE
	# 0007, WRITE
0004	LD, CNT, 00, WRITE
0005	OUT, 0100, WRITE
0006	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

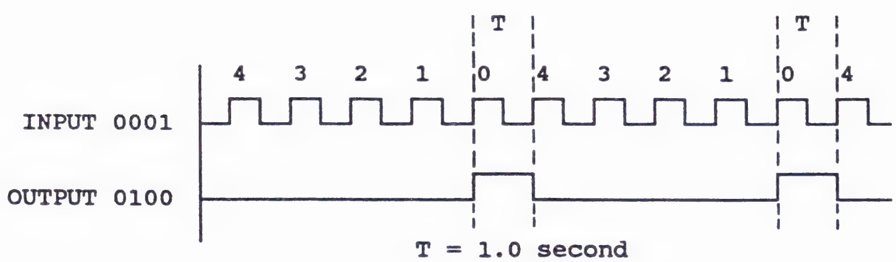
Figure 4.37 Counter Used As a Retentive Timer

After entering the program, debug, run, and monitor the program.

Self-Resetting Counter

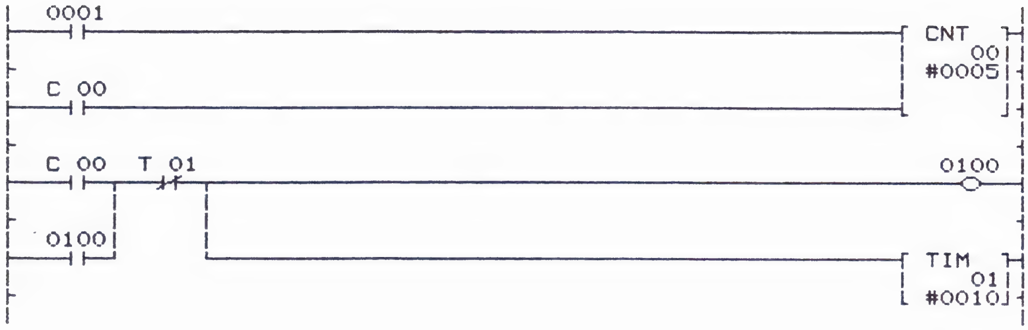
Figure 4.38 depicts a Timing Chart, Ladder Diagram, and Statement Logic for a counter that resets itself.

Timing Chart



Output 0100 will energize for 1 second after every 5 counts.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	LD, CNT, 00, WRITE
0002	CNT, 00, WRITE
	# 0005, WRITE
0003	LD, CNT, 00, WRITE
0004	OR, 0100, WRITE
0005	AND, NOT, TIM, 01, WRITE
0006	OUT, 0100, WRITE
0007	TIM,, 01, WRITE
	# 0010, WRITE
0008	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.38 Self Resetting Counter

After entering and verifying this example, debug, run, and monitor the program.

FUN 12 (Reversible Counter)

FUN 12 (CNTR) signifies a Reversible (Up-Down) Counter. It increases or decreases the count value by one (1) whenever the UP signal or DOWN signal goes from OFF to ON. When both the UP and DOWN signals are turned ON simultaneously, no counting is done.

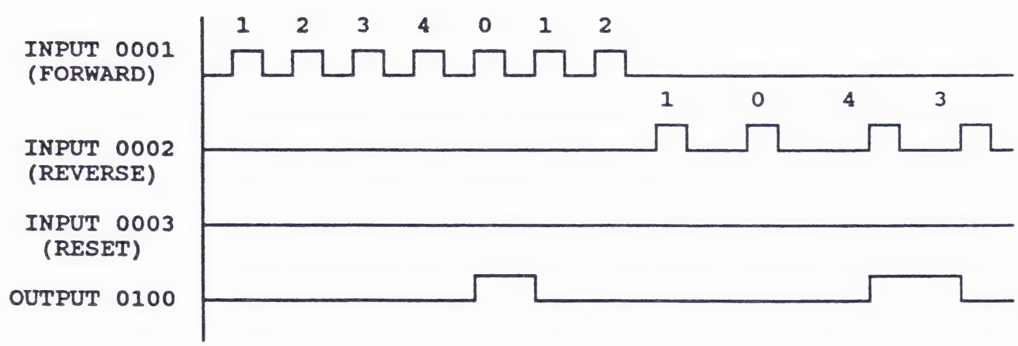
CNTR is also known as a Ring Counter, meaning that if decreased from 0000, it goes to the preset value. Therefore, the output is turned ON only when the count value has been increased or decreased to the preset value.

The Reset signal is accepted both during count-up and count-down. While it is ON, the present value is reset to "0000" and neither the UP or DOWN signal is accepted.

Reversible Counters are typically used in position-type control applications where the preset value determines the boundaries of a position.

Figure 4.39 depicts an example of a Reversible Counter including a Timing Chart and Ladder Diagram.

Timing Chart



Ladder Diagram

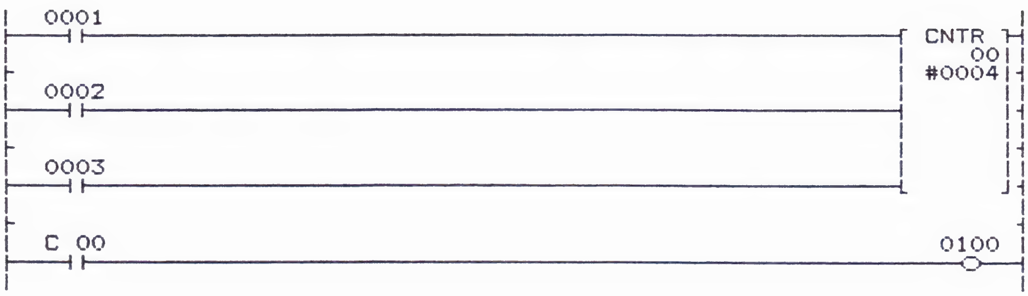


Figure 4.39 Reversible Counter

Changing Preset and Present Counter Values; Set/Reset

The procedures for changing preset and present counter values as well as forcing (setting and resetting) counters are exactly the same as those for timers. See pages 141 to 143 for complete step-by-step instructions. Merely substitute the appropriate counter number where a timer is indicated.

External Setting of Timers and Counters

The preset value for both Timers and Counters can be set by an external setting device. The preset value for a Timer/Counter will then be a channel location where the setting device (BCD thumbwheel, etc.) is wired to.

Figure 4.40 shows a BCD thumbwheel wired directly to Channel 02 (0200-0215).

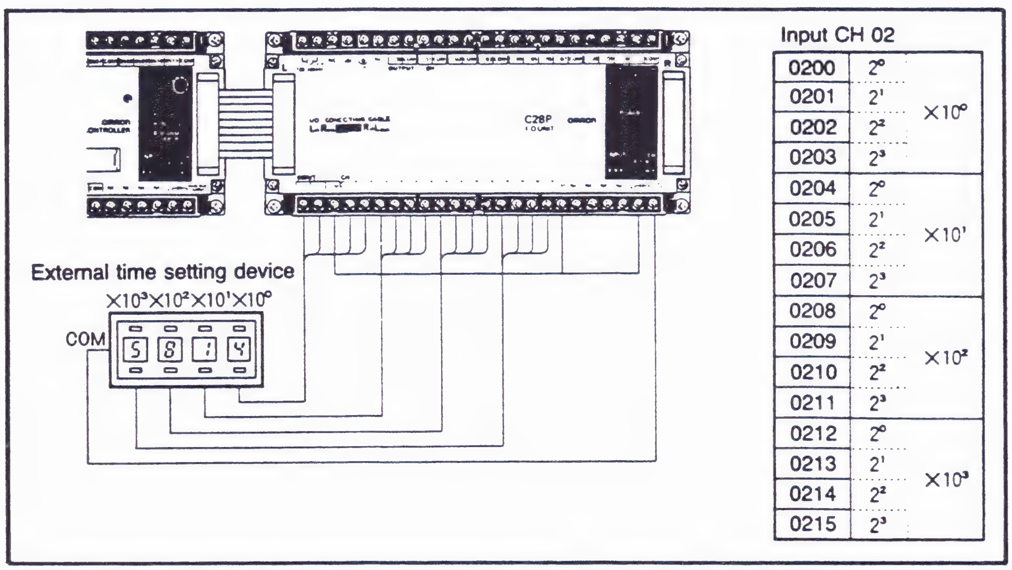


Figure 4.40 Wiring of External Time Setting Device

NOTE

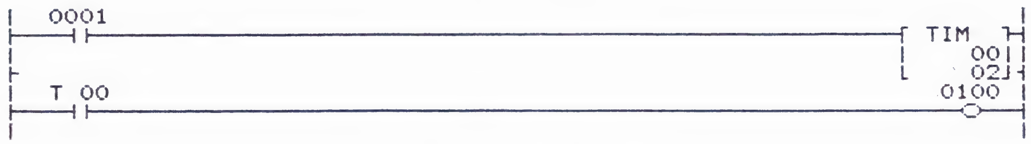
The externally set time value must be in 4-digit Binary-Coded Decimal (BCD) from 0 to 9999; otherwise, an error occurs and the Error Flag (Special Auxiliary Relay 1903) is turned ON. At this time, channel data not in BCD are ignored completely, and the time value remains the same.

Timer/Counter values may be specified in the following areas:

I/O, INTERNAL AUXILIARY RELAY	00 to 17
HOLDING RELAY	0 to 9

Figure 4.41 depicts an example of a timer using a channel location as a preset value.

Ladder Diagram



Statement Logic

Address	Key Sequence
0000	LD, 0001, WRITE
0001	TIM, 00, WRITE

[A 2-digit channel number must be specified as the time value in the program. To do this, press the CLR key after specifying the timer coil. Then specify the channel number followed by the WRITE key]

	CLR, 02, WRITE
0002	LD, TIM, 00, WRITE
0003	OUT, 0100, WRITE
0004	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.41 Program with Externally Set Time Value

FUN 20 (Compare)

This function signifies a Compare (CMP) statement. It is used to compare the data in a specific channel with the data in another channel, or a 4-digit hexadecimal constant. Two sets of data must therefore be specified immediately after the CMP statement, one of which must be a channel.

FUN 20 then, compares the numbers in two channels and initiates the instructions.

There are three auxiliary relays that are an integral part of the Compare statement:

(1) Relay 1905 - greater than (>)
(2) Relay 1906 - equal to (=)
(3) Relay 1907 - less than (<)

When calling out a Compare statement, two groups of data are required as follows:

(1) Data^1 to be compared (2) Data^2 to be compared	Either one may be termed Data A or Data B

The following channels or constants may be specified:

I/O, INTERNAL AUXILIARY RELAY	00 to 17
SPECIAL AUXILIARY RELAY	18 to 19
HOLDING RELAY	0 to 9
TIM/CNTs	00 to 47
CONSTANTS	0000 to FFFF
DATA MEMORY	00 to 63

The relationship between Compare Flags 1905, 1906, and 1907, and the FUN 20 instruction is as follows:

(a) Relay 1905 equals a "1" condition if Data A is greater than Data B.
(b) Relay 1906 equals a "1" condition if Data A is equal to Data B.
(c) Relay 1907 equals a "1" condition if Data A is less than Data B.

Figure 4.42 shows a program example using a Compare statement with the three Compare Flags. In this example, data in HR Channel 0 (Data A) will be compared to the constant #0250 (Data B). Before entering this program into the PLC, remember to erase the previous program (p.76).

<u>Ladder Diagram</u>

<u>Statement Logic</u>

<u>Address</u>	<u>Key Sequence</u>
0000	LD, 0001, WRITE
0001	OUT, TR, 0, WRITE
0002	FUN, 20, WRITE
	(a) HR, 0, WRITE
	(b) # 0250, WRITE
0003	LD, TR. 0. WRITE
0004	AND, 1905, WRITE
0005	OUT, 0100, WRITE
0006	LD, TR, 0, WRITE
0007	AND, 1906, WRITE
0008	OUT, 0101, WRITE
0009	LD, TR, 0, WRITE
0010	AND, 1907, WRITE
0011	OUT, 0102, WRITE
0012	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.42 Compare Statement

After entering and verifying the program, perform the following:

(1) Switch the Programming Console to the MONITOR mode.

(2) Press the CLR key to clear the display.

(3) Monitor Channel HR0 by pressing:

SHIFT, CH/*, HR, 0, MONTR

The display should show:

```
c H0
0000
```

This indicates the current value of HR0 (all 16 points) is 0000.

(4) To enter a value (constant #) for HR0, press the CHG key. The display should show:

```
0000PRES VAL?
c H0  0000  ????
```

Channel which is monitored	Present value	Where new value is entered

(5) Enter the value: 0200, WRITE.
The display shows:

```
c H0
0200
```

(6) By energizing Input 0001 (the input that commands the PLC to perform the Compare statement), Output 0102 should energize because:

Data A = HR0, 0200
Data B = 0250

Since Data A is less than Data B, the Compare Flag 1907 turns ON and energizes Output 0102.

(7) Repeat Steps 4 and 5, but enter the value 1000 in HR0. In this case, after the value is entered and the Compare statement is performed (Input 0001 is energized), Output 0101 will turn ON because:

Data A = 1000
Data B = 0250

Since Data A is greater than Data B, Compare Flag 1905 turns ON and energizes Output 0100.

(8) Now repeat the instructions in Step 7 but enter the value 0250 in HR0.

NOTE

The Compare instruction is executed once every time the CPU scans the program. Use a DIF statement (FUN 13, 14) to execute only once.

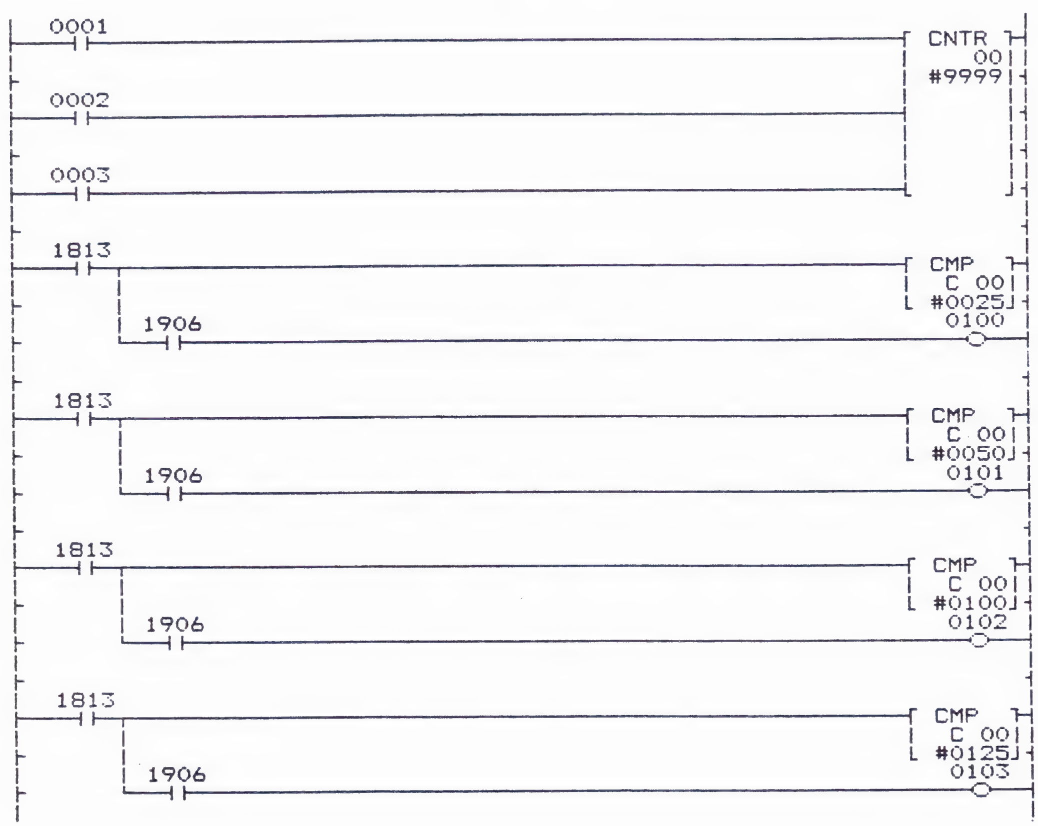
Application Example

Instead of using multiple counters, the program example in Figure 4.43 will utilize one (1) Reversible Counter (FUN 12) and four (4) Compare (FUN 20) statements to allow for 4 separate presets with 4 outputs. The following requirements must be met:

(a) Output 0100 will energize when Input 0001 energizes 25 times.
(b) Output 0101 will energize when Input 0001 energizes 50 times.
(c) Output 0102 will energize when Input 0001 energizes 100 times.
(d) Output 0203 will energize when Input 0001 energizes 125 times.

Before entering this program into the PLC, erase the previous program (p.76).

Ladder Diagram



NOTE

Contact 1813 is always a closed contact.

Statement Logic

Address	Key Sequence	Address	Key Sequence
0000	LD, 0001, WRITE	0013	LD, TR, 0, WRITE
0001	LD, 0002, WRITE	0014	AND, 1906, WRITE
0002	LD, 0003, WRITE	0015	OUT, 0101, WRITE
0003	FUN, 12, 00, WRITE	0016	LD, 1813, WRITE
	#9999, WRITE	0017	OUT, TR, 0, WRITE
0004	LD, 1813, WRITE	0018	FUN, 20, WRITE
0005	OUT, TR, 0, WRITE		(a) CNT, 00, WRITE
0006	FUN, 20, WRITE		(b) #0100, WRITE
	(a) CNT, 00, WRITE	0019	LD, TR, 0, WRITE
	(b) #0025, WRITE	0020	AND, 1906, WRITE
0007	LD, TR, 0, WRITE	0021	OUT, 0102,WRITE
0008	AND, 1906, WRITE	0022	LD, 1813, WRITE
0009	OUT, 0100, WRITE	0023	OUT, TR, 0, WRITE
0010	LD, 1813, WRITE	0024	FUN, 20, WRITE
0011	OUT, TR, 0, WRITE		(a) CNT, 00, WRITE
0012	FUN, 20, WRITE		(b) #0125, WRITE
	(a) CNT, 00, WRITE	0025	LD, TR, 0, WRITE
	(b) #0050, WRITE	0026	AND, 1906, WRITE
		0027	OUT, 0103, WRITE
		0028	FUN, 01, WRITE

To verify press: CLR, CLR, SRCH.

Figure 4.43 Compare Statements Used with a Reversible Counter

Exercise

Change the preset values for the counters in Figure 4.43.

4.10 PROGRAM STORAGE

Once the operator has edited the program and completed a test run, he/she may wish to make a permanent copy for future reference or program back-up. Two methods of program storage are available:

(a) Cassette Tape Transfer.
(b) Storing onto an EEPROM memory chip.

4.10.1 Cassette Tape Transfer

User programs may be effectively stored by recording them onto cassette audio tape using a commercially available monaural cassette tape recorder (Figure 4.44).

The following hardware is needed:

(a) Programming Console
(b) Connecting Cable (Omron SCYPOR-PLG01 or equivalent non-audio style cabling)
(c) Monaural Cassette Tape Recorder
(d) Cassette Tape (120 minutes or less, but at least 7 minutes long)

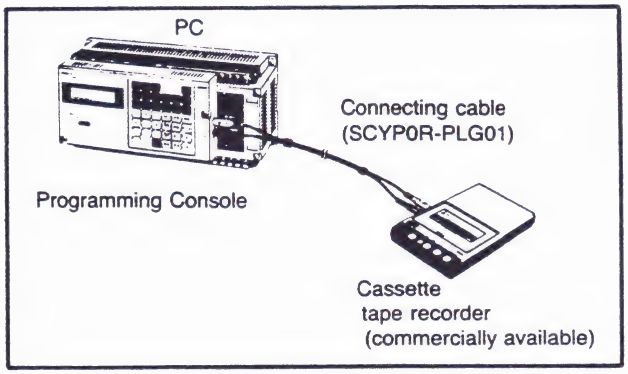


Figure 4.44 Program Transfer Between PLC and Cassette Tape

The following paragraphs describe three modes of storage:

(a) Tape Write (Record) (PLC to Cassette)
(b) Tape Play (Cassette to PLC)
(c) Tape Verify (used to verify both operations)

4.10.1.1 Tape Record (PLC to Cassette Tape)

Before recording, the operator must choose an 8-digit file number which serves as an identification number for each program to be stored. The file number also prevents the downloading and uploading of a program by unauthorized personnel.

Pre-Tape Connections

(1) Connect the patchcord between the MIC jack on the Programming Console and the MIC (or Line-In) jack on the tape recorder.

(2) Connect a second cord between the EAR (or Line-Out of a tape recorder) jacks of both devices. (There is no need to transpose connections, as it is done internally in the Programming Console.)

Recording Operation

(1) Set the 3-position slide switch on the Programming Console to the PROGRAM mode.

(2) Turn the Volume and Tone controls of the tape recorder to their maximum levels.

(3) Press the CLR key on the Programming Console twice.

(4) Press the EXT key.

(5) Enter the chosen 8-digit file number (see above).

(6) Press the SHIFT key.

(7) Press the appropriate button(s) on the tape recorder to begin the recording process.

(8) 5 seconds after performing Step 7, press the REC/RESET key on the Programming Console. This allows the blank leader of a standard tape to pass the heads.

At this point, a blinking rectangle should appear in the right-hand corner of the LCD display. This indicates the program is being recorded. The entire process takes

approximately 7 minutes. After the complete RAM memory has been recorded, the LCD will show the last RAM address and the message RECORD END.

4.10.1.2 Program Loading (Cassette to PLC)

The following operation transfers the program data recorded onto cassette tape into the RAM user memory in the CPU. Perform the same pre-tape connections and ensure the Programming Console is in the PROGRAM mode as indicated in Paragraph 4.10.1.1. Rewind the tape so that a few seconds of blank tape precede the beginning of the program, and perform the following steps in sequence:

(1) Press the EXT key.

(2) Enter the appropriate 8-digit number.

(3) Press the SHIFT key.

(4) Start the tape recording by pressing the PLAY button.

(5) Press the PLAY/SET key on the Programming Console. The display should show:

```
0000MT PLAY        ■
```

and the blinking rectangle will appear in the upper right.

(6) When the loading is complete (approximately 7 minutes), the display will read:

```
XXXXPLAY END
```

4.10.1.3 Program Verification

After completing the Program Load operation, it is always best to confirm that the data has been transferred properly from the tape to the RAM.

To verify the program, perform the following:

(1) Rewind the tape and provide about 5 seconds of blank tape leader before the taped program begins.

(2) Make sure the Programming Console is set to the PROGRAM mode.

(3) Press the CLR key twice, and then the EXT key.

(4) Enter the appropriate 8-digit file number.

(5) Press the PLAY button on the tape recorder.

(6) Press the VER key on the Programming Console. The blinking rectangle should appear in the right-hand corner of the LED indicating that program verification is taking place.

(7) After approximately 7 minutes, VER END should be displayed indicating the contents in the PLC verify the contents of the tape. If an error message appears on the LCD, repeat Steps 3 through 6.

4.10.2 P-ROM Storage

Another way to store and retrieve programs is by using the P-ROM Writer. Unlike the cassette tape method, P-ROM storage allows the user to save a program to a working portion of the system.

By using an Eraseable Programmable Read-Only Memory (EPROM) chip, the user ensures non-volatile storage for his/her program and avoids the need for a battery back-up.

Omron employs the Ultraviolet (UV) style of EPROM chip which can only be erased by exposure to an ultraviolet light source for a certain amount of time.

The following paragraphs outline the hardware and system set-up requirements needed to perform this program support function which includes the ERASE check, WRITE, READ, and VERIFY operations. For more detailed information, see the P-ROM Writer Operation Guide, Omron publication W108-E1-1.

4.10.2.1 Hardware

The Omron P-ROM Writer (3G2A5-PRW05-E-V1) supports the complete line of Omron C-Series group of Programmable Controllers. As shown in Figure 4.45, the P-ROM Writer may be mounted directly to the PLC or on the GPC to perform READ/WRITE functions.

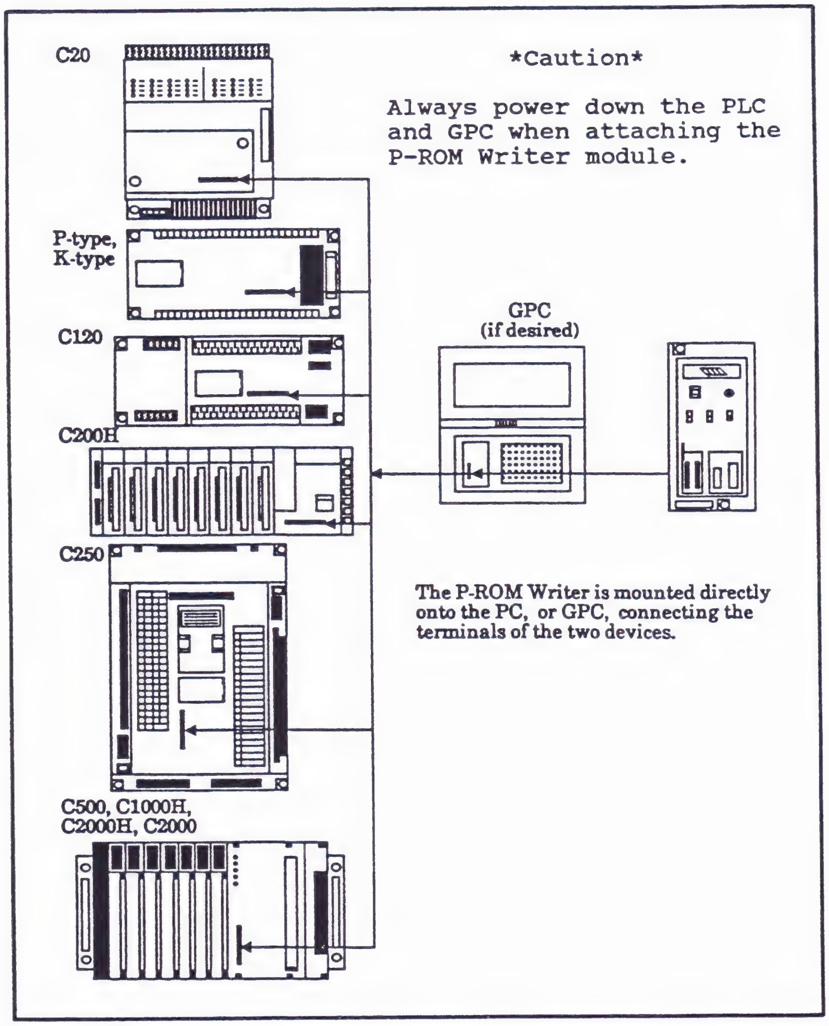


Figure 4.45 Mounting the P-ROM Writer

The P-ROM Writer itself conforms in size to all the other types of peripheral support devices (Programming Console, Printer Interface, etc.) and as shown in Figure 4.45 may be

mounted directly to the peripheral interface connector of the chosen PLC.

Figure 4.46 illustrates the layout of the front panel of the P-ROM Writer which is exposed to the user during operation.

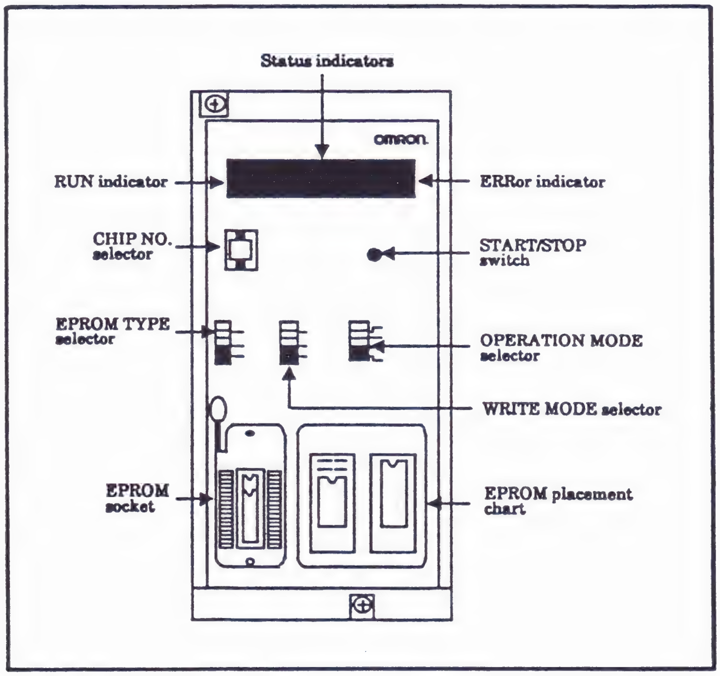


Figure 4.46 Front Panel of P-ROM Writer

The settings are described as follows:

(1) Status Indicators - 6 LEDs are located behind the viewer panel for RUN-STATUS-ERROR indication.

(2) START/STOP Switch - used to Start/Stop P-ROM Writer operation.

(3) 4-position OPERATION MODE Selector:

(a) Read - Reads contents of EPROM chip to user memory (RAM) in PLC or GPC.

(b) Write - Writes user memory in PLC or GPC to EPROM chip.

(c) Verify - Verifies (checks) contents of user memory against program in device (PLC or GPC).

(d) Erase Check - Verifies that EPROM chip is erased before performing write functions.

NOTE

EPROM chip must be totally erased before performing a WRITE operation.

(4) EPROM Selector - Used to preset which EPROM chip is being read from or written to.

(5) WRITE MODE Selector - used to preset the speed of a WRITE operation (applicable to which EPROM chip is chosen).

(6) CHIP NO. Selector Thumbwheel - used to select the appropriate chip number when writing a program to the chip(s). For example, the C2000H PLC may need 4 separate EPROM chips to store the complete user memory. (See Page 7 of the P-ROM Writer Operation Guide, Cat. No. W108-E1-1 for further description and chip guidelines).

(7) EPROM Socket - Secures the EPROM chip to be written to or read from.

Figure 4.47 shows the Back Panel and appropriate PLC setting configurations. Note thal all dipswitches are preset in the factory to the OFF position.

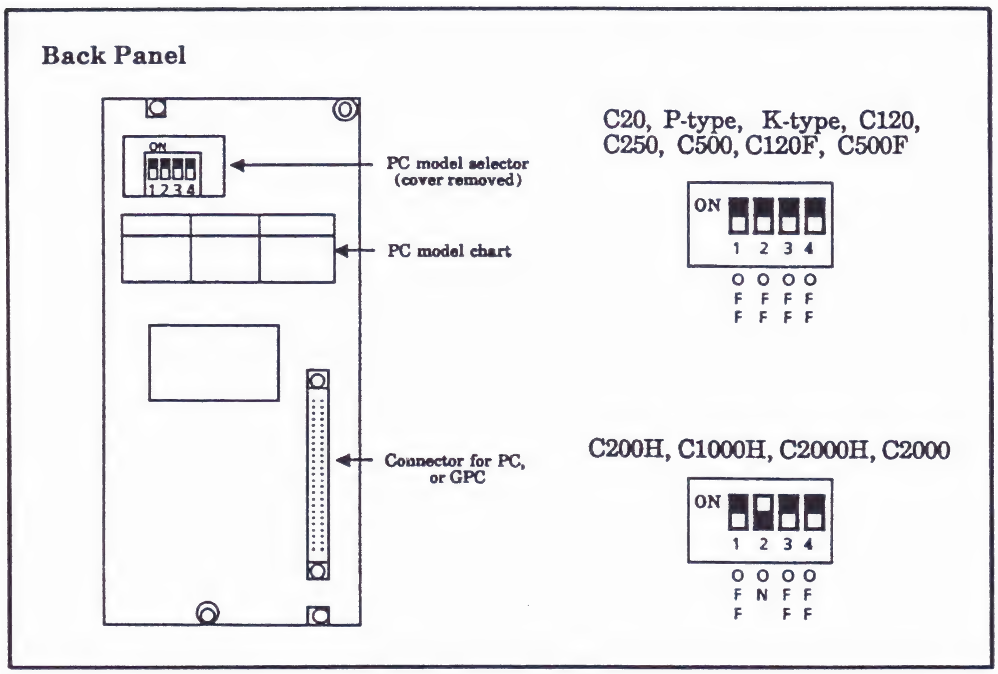
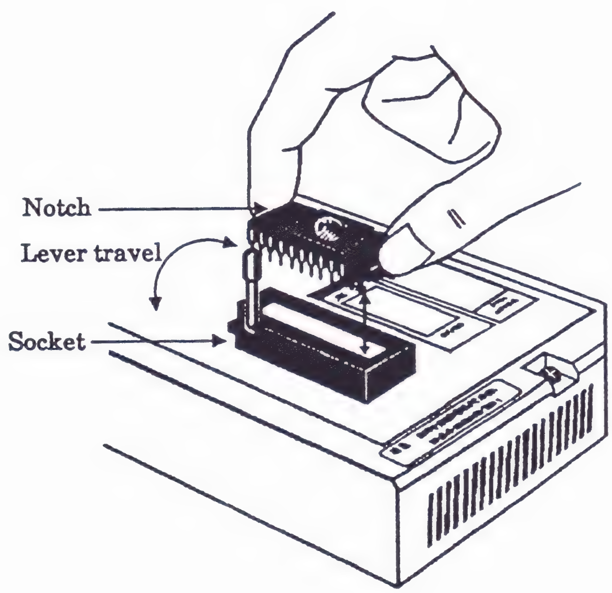


Figure 4.47 Back Panel and Switch Settings

4.10.2.2 EPROM Chip Installation

After making the correct switch settings per specific PLC and chip requirements, attach the P-ROM Writer unit to the desired device (PLC or GPC). Install the appropriate chip to the P-ROM Writer by following the instructions in Figure 4.48.



1. Raise the lever to unlock the socket.
2. Holding the chip so that you do not touch the pins, insert it into the socket, with the notch end toward the top of the P-ROM Writer, as shown in the EP-ROM placement chart. Note that in the case of ROM-GA (2732A equivalent), there are only 24 pins, leaving the top terminals of the socket open as shown in the chart on the front panel of the P-ROM Writer.
3. After confirming the proper insertion of the chip, lower the lever to lock it in place.

Figure 4.48 Installing the EPROM Chip

4.10.2.3 P-ROM Writer Function Operations

The following paragraphs cover P-ROM Writer operation. Before writing to the EPROM chip, it is necessary to first perform an Erase Check to verify that the chip is void of user instructions.

<u>Erase Check</u>

(1) Set the EPROM TYPE selector to the type of chip being tested.

(2) Set the OPERATION MODE selector switch to ERASE CHECK.

(3) Press the START/STOP button to begin.

If the chip is completely erased, no indicator lights will illuminate on the P-ROM Writer Status Display. If however, the ERROR indicator illuminates during the check, the chip is not completely erased, and cannot be written to until it has been entirely erased.

To interpret the LED indicators refer to Table 4.2 below:

Table 4.2 Status Display Indication for Erase Check

	RUN	STATUS	ERROR
Checking	○	●●●●	●
Erased	●	●●●●	●
Unerased	●	●●●●	○

○: lit ●: unlit

Note: The status indicator does not function during erase check.

See Table 4.6 for causes and correction of error indications.

<u>Writing to EPROM (GPC or PLC User Memory to EPROM Chip)</u>

Write to the chip only after performing the Erase Check.

Perform the following:

(1) Set the EPROM TYPE selector and CHIP NO. selector switches to reflect the type of chip being written.

(2) Reconfirm the Dip Switch settings on the back panel to reflect the type of PLC used.

(3) Set the WRITE MODE selector switch to the appropriate setting according to which chip is being used.

(4) Set the OPERATION MODE selector switch to WRITE.

(5) Press the START/STOP button to begin. (Refer to page 15 of the P-ROM Writer Operation Guide (W108-E1-1) for processing time values.)

See Table 4.3 below to interpret the Status LED indicators on the P-ROM Writer.

Table 4.3 Status Display Indications for WRITE Operation

	RUN	STATUS	ERROR
Beginning	○	◐○○○	●
1/4 Complete	○	●◐○○	●
1/2 Complete	○	●●◐○	●
3/4 Complete	○	●●●◐	●
Complete	●	●●●●	●
Error	●	●●●●	○

○: lit ◐: blinking ●: unlit

Refer to Table 4.6 for causes and corrections of Error indications.

Reading from EPROM (EPROM chip to PLC and GPC RAM Memory)

Perform the following:

(1) Follow Steps 1 through 3 of the "Writing to EPROM" operation.

(2) Set the OPERATION MODE selector switch to READ.

(3) Press the START/STOP button to begin the Read operation.

See Table 4.4 below to interpret the Status LED indicators on the P-ROM Writer.

Table 4.4 Status Display Indicators for Read Operation

	RUN	STATUS	ERROR
Reading	○	●●●●	●
Complete	●	●●●●	●
Error	●	●●●●	○

○: lit ●: unlit

Note: The status indicator does not function while reading.

Refer to Table 4.6 for causes and correction of error indications.

Verification (EPROM to RAM and Vice Versa)

After performing a READ or WRITE operation, always verify the contents of the EPROM chip against the RAM user memory in the PLC or GPC.

To verify, follow the steps below:

(1) Follow Steps 1 through 3 of the "Writing to EPROM" operation.

(2) Set the OPERATION MODE selector switch to VERIFY.

(3) Press the START/STOP button to begin verification.

See Table 4.5 below to interpret the Status LED indicators on the P-ROM Writer.

Table 4.5 Status Display Indications for Verify Operation

	RUN	STATUS	ERROR
Verifying	○	●●●●	●
Complete	●	●●●●	●
Error	●	●●●●	○

○: lit ●: unlit

Note: The status indicator does not function during verification.

See Table 4.6 for causes and correction of Error indications.

Troubleshooting

If the ERROR indicator illuminates during one of the operations discussed above, press the START/STOP button to release the error. Refer to Table 4.6 to determine and correct possible causes of Read, Write, and Verify errors.

Table 4.6 Table of Errors

Mode	Possible Cause	Correction
ERASE CHECK	EPROM chip is not empty.	Use an empty chip.
WRITE VERIFY READ	EPROM TYPE selector, DIP switch, or CHIP NO. selector not set properly.	Check, and correct all the switch settings.
	Chip type incompatible with SYSMAC model.	Exchange with appropriate chip type.
WRITE	EPROM chip is unerased, or defective.	Replace the EPROM chip.
VERIFY	The contents of the memory and the chip do not match.	After WRITE: Erase the chip, and write it again.
		After READ: Read the chip to the memory again.
READ	A ROM chip is installed in the PC.	Exchange it with a RAM chip.
	The EPROM chip is not written.	Exchange it with a chip that is written.
	The Host Link mode switch is set to "HOST" mode.	Set the switch to "LOCAL" mode.

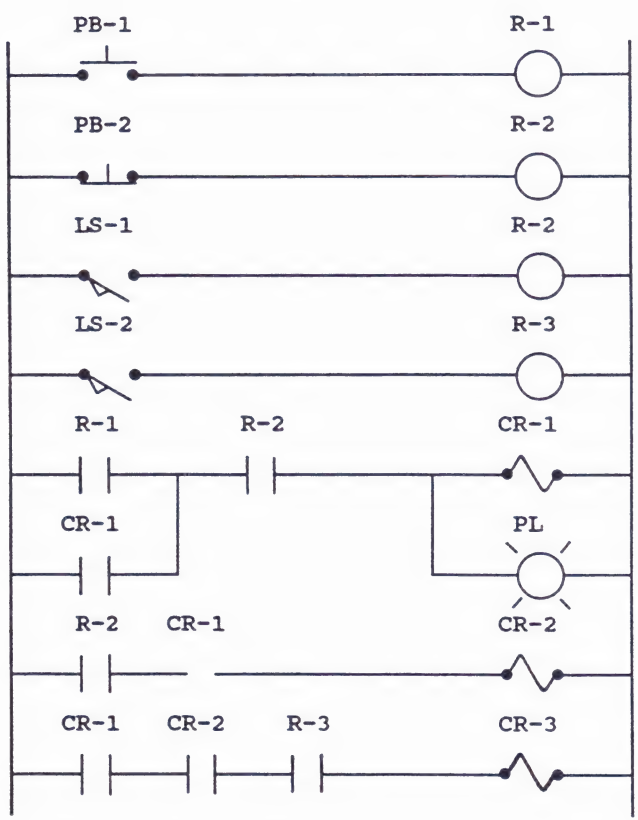
4.11 APPLICATION EXERCISES

The following application exercises are provided to strengthen the user's skills in converting hard-wired logic to PLC Ladder Logic and Statement Logic, and reinforce work with Timing Charts. Lastly, the user is presented with a control scenario featuring an Automatic Liquid Pouring System for which a complete program must be created.

Exercise No. 1

Convert the hard-wired logic presented below to PLC Ladder Logic and prepare the Statement Logic to be entered into the PLC.

Hard-wired Logic



Ladder Logic

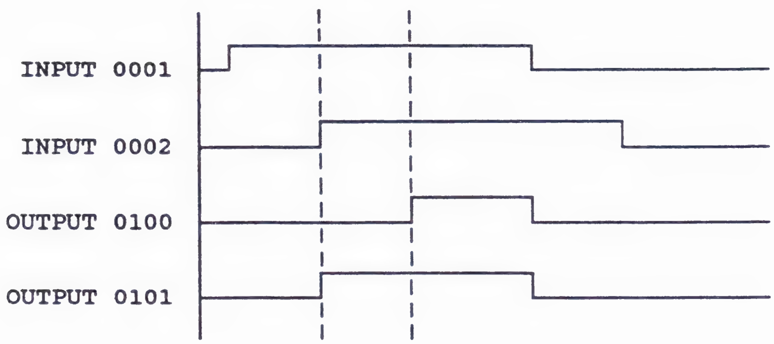
Statement Logic

Address	Key Sequence

Exercise No. 2

Study the timing chart below and write a program for it.

Timing Chart



Ladder Diagram

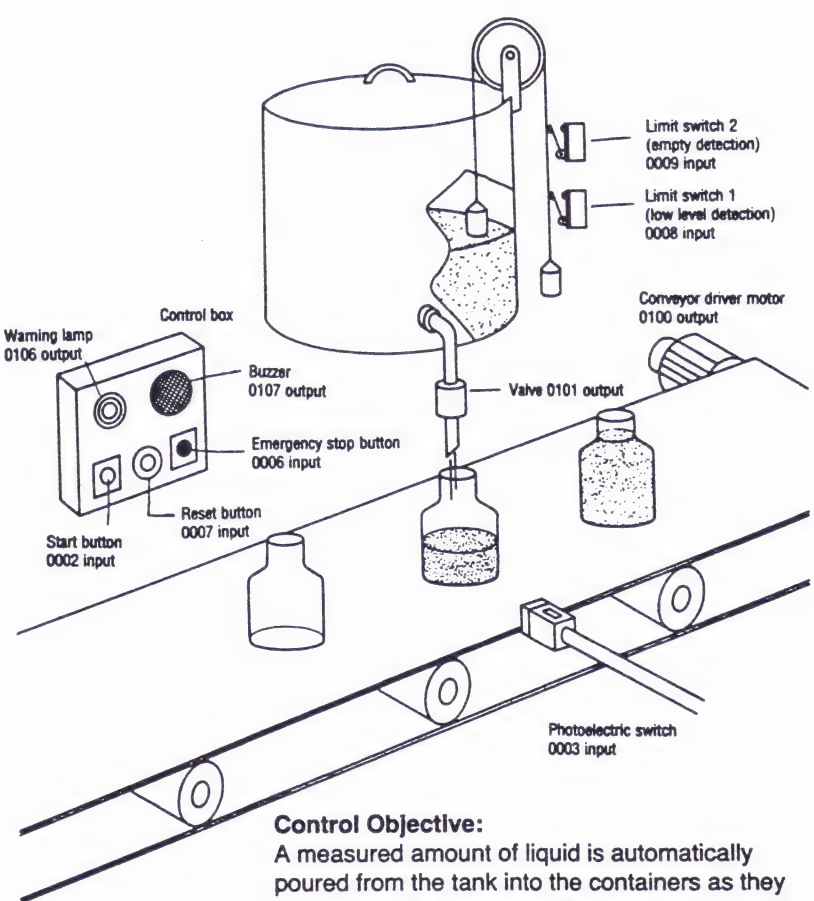
Statement Logic

Address	Key Sequence

Exercise No. 3

Study the control scenario depicting an Automatic Liquid Pouring System.

Automatic Liquid Pouring System



Control Objective:
A measured amount of liquid is automatically poured from the tank into the containers as they move along the conveyor. The warning lamp will flash when the fluid in the tank reaches a minimum. When there is no fluid left in the tank, the buzzer will sound and the drive motor will stop.

Note the following criteria:

(1) <u>Inputs</u>

0002 - Start pushbutton
0003 - Photoelelectric switch
0006 - Stop pushbutton
0007 - Error Reset pushbutton
0008 - Limit Switch 1
(Low level detection)
0009 - Limit Switch 2
(Empty detection)

<u>Outputs</u>

0100 - Conveyor mode
0101 - Valve
0106 - Warning Lamp
(Low level indicator)
0107 - Buzzer
(Empty indicator)

(2) Fill is premeasured in terms of time (1.3 seconds).

(3) Low level lamp flashes ON/OFF.

(4) Buzzer sounds and system shuts down when an empty condition occurs.

After considering all the criteria, create a Ladder Diagram, write a program, load the Statement Logic into the PLC, debug the program, and lastly run the program.

Ladder Diagram

Statement Logic

Address	Key Sequence

For Information Call:

1-708-843-7900

OMRON ELECTRONICS, INC.
Training Center
One East Commerce Drive
Schaumburg, IL 60173
FAX (708) 843-7787